U.PORTO

MESTRADO
MULTIMÉDIA - ESPECIALIZAÇÃO EM TECNOLOGIAS

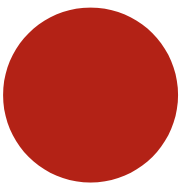
EDIÇÕES CRÍTICAS DIGITAIS: O CASO DA LITERATURA ELECTRÓNICA DE PEDRO BARBOSA

Carlos Filipe Lopes do Amaral

M
2015

FACULDADES PARTICIPANTES:

FACULDADE DE ENGENHARIA
FACULDADE DE BELAS ARTES
FACULDADE DE CIÊNCIAS
FACULDADE DE ECONOMIA
FACULDADE DE LETRAS



U.PORTO

Edições Críticas Digitais: O Caso da Literatura Electrónica de Pedro Barbosa

Carlos Filipe Lopes do Amaral

Mestrado em Multimédia da Universidade do Porto

Orientador: Rui Torres (Professor Associado com Agregação da Faculdade de Ciências Humanas e Sociais da Universidade Fernando Pessoa)

Co-orientador: João Correia Lopes (Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto)

31 de Julho de 2015

Edições Críticas Digitais: O Caso da Literatura Electrónica de Pedro Barbosa

Carlos Filipe Lopes do Amaral

Mestrado em Multimédia da Universidade do Porto

Aprovado em provas públicas pelo Júri:

Presidente: Nuno Flores (Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto)

Vogal Externo: José Manuel Torres (Professor Associado da Faculdade de Ciências e Tecnologia da Universidade Fernando Pessoa)

Orientador: Rui Torres (Professor Associado, com Agregação, da Faculdade de Ciências Humanas e Sociais da Universidade Fernando Pessoa)

Resumo

As edições críticas digitais são colecções de obras nascidas no seio digital, e operam como arquivos digitais — contemplando a interactividade e o comentário crítico.

A literatura electrónica rege-se por uma multitude de práticas literárias que tira partido das capacidades e contextos proporcionados pelo computador — consequentemente, é o resultado ou produto da actividade literária realizada no computador.

Pedro Barbosa, autor de literatura gerada por computador, é uma figura central no campo da literatura electrónica e na produção de obras de arte nascidas no meio digital especialmente pela sua presença remontar aos inícios da computação até às práticas de criação artística no computador vigentes na conjuntura tecnológica contemporânea.

A importância de estabelecer uma relação entre a literatura electrónica e edições críticas digitais é o ponto central da dissertação. O desenvolvimento de uma Aplicação Web como modelo de edição crítica digital que vise representar, explicar e compilar as obras digitais de literatura electrónica de Pedro Barbosa é o principal objectivo deste estudo, onde se inclui código, *software*, geradores textuais e comentário crítico.

No Capítulo *Humanidades na Computação* é mapeado um percurso cronológico do envolvimento das Humanidades no aparecimento do computador e das linguagens de programação com que Pedro Barbosa esteve em contacto durante quase meio século para a produção de artefactos literários. Os trabalhos pioneiros de Silvestre Pestana, bpNichol, Nanni Balestrini, entre outros, servem de exemplo para caracterizar o vasto espólio de géneros literários que foram criados, e literariamente criticados e investigados.

No Capítulo *Especificação da Aplicação Web* são reunidos requisitos técnicos, e é desenhada a arquitectura lógica e implementada a Aplicação Web "Pedro Barbosa: Edição Crítica Digital", como plataforma-modelo especialmente criada para a representação de obras de literatura electrónica.

Um modelo de edição crítica digital intitulada *Pedro Barbosa: Edição Crítica Digital* foi desenvolvido para representar o espólio de artefactos digitais de Pedro Barbosa. Doravante haverá um continuado esforço na criação de novas funcionalidades com o intuito de enriquecer o modelo de edição crítica digital.

Palavras-chave: literatura electrónica, arquivo digital, poesia experimental, linguagens de programação, obsolescência tecnológica, edição crítica digital, humanidades.

Abstract

Critical digital editions are collections art born digital, and operate as digital archives — contemplating the interactivity and critical thinking.

Electronic literature is a multitude of literary practices that take advantage of the capabilities and settings provided by the computer — consequently, it is the result or product of literary activity on the computer.

Pedro Barbosa, author of literature generated by the computer, is a central figure in the field of electronic literature and the production of works of art born in the digital environment, especially by taking into account his presence from the beginnings of computing to the artistic creation practices of the computer environment in contemporary technology.

The importance of establishing a relationship between the electronic literature and critical digital editions is the focus of the dissertation. Developing a Web Application as a critical critical edition model that aims to represent, explain and compile digital works of electronic literature by Pedro Barbosa is the main aim of this study where it is contemplated code, software, text generators and critical comment.

In Chapter *Humanities in Computation* it is mapped a chronological journey of involvement of the Humanities in the computer and the appearance of programming languages in which Pedro Barbosa has been in contact for nearly half a century for the production of literary artifacts. The pioneering work of Silvestre Pestana, bpNichol, Nanni Balestrini, amongst others, serve as examples to characterize the vast collection of literary genres that were created, and literary criticitized and investigated.

In Chapter *Specification of the Web Application* are gathered technical requirements, it is drawn the arquitecture's logical view and the Web Application entitled *Pedro Barbosa: Digital Critical Edition* is implemented as a model platform especially created for the representation of works of electronic literature.

A critical digital edition model "Pedro Barbosa : Digital Critical Edition" was developed to represent Pedro Barbosa's digital artifacts. In the future there will be a continued effort towards the creation of new features to enrich the model of a critical digital edition.

Keywords: electronic literature, digital archive, experimental poetry, programming languages, technological obsolescence, critical digital edition, humanities

Agradecimentos

Finda a dissertação de Mestrado em Multimédia, cabe-me agradecer aos docentes que me acompanharam neste percurso: Professor Doutor Rui Torres (orientador) e Professor Doutor João Lopes Correia (co-orientador). Do mesmo modo aos meus pais por me terem apoiado financeiramente. Por último, ao Tiago Azevedo, finalista do MIEIC, e aos programadores do *livechat* oficial do Laravel.

Carlos Amaral

Notas

Esta dissertação não foi escrita ao abrigo do novo Acordo Ortográfico.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ACH	Association for Computers and the Humanities
ALLC	Association for Literary and Linguistic Computing
CETIC	Centro de Estudos sobre Texto Informático e Ciberliteratura
CIUP	Centro de Informática da Universidade do Porto
CODASYL	Committee on Data Systems Languages
E-lit.	Literatura Electrónica ou Electronic Literature
ELO	Electronic Literature Organization
ETH	Federal Institute of Technology
FCUP	Faculdade de Ciências da Universidade do Porto
FEUP	Faculdade de Engenharia da Universidade do Porto
HTML	HyperText Markup Language
ICCH	International Conference on Computing in the Humanities
ICL	International Computers Limited
JS	JavaScript
JVM	Java Virtual Machine
LACA	Laboratório de Cálculo Automático
MVC	Model-View-Controller
OTA	Oxford Text Archive
SGML	Standard Generalized Markup Language
SO	Sistema Operativo
SVG	Scalable Vector Graphics
TEI	Text Encoding Initiative
UFP	Universidade Fernando Pessoa
WWW	World Wide Web

Capítulo 1

Introdução

Contrariamente aos arquivos digitais, que operam como um espólio de obras digitais, as edições críticas digitais consideram de demais importância a inclusão da interpretação; o utilizador é mais do que um espectador passivo, faz uso da possibilidade de ter um papel activo e ser um interpretador.

As edições críticas digitais assumem especial relevo na transição da Web 2.0 para a 3.0, pelas capacidades que as tecnologias existentes proporcionam na concepção de aplicações Web. De facto, ao modelos de edições críticas digitais, permitem níveis de interactividade e participação colaborativa mais ricos — proporcionando uma melhor experiência ao nível de utilização. A curadoria digital comporta um papel fundamental na organização de obras digitais.

A literatura electrónica assume, actualmente, uma ambivalência de significado pela união entre a tradição literária e as transformações que têm vindo a ser introduzidas pelo advento da computação. Não obstante, os leitores entram em contacto com obras de arte digitais, comportando expectativas inerentes à literatura física e seus cânones. Contudo, a literatura electrónica deverá ser criada no suporte destas expectativas mas deverá, fundamentalmente, modificá-las e transformá-las para que um novo paradigma se afirme.

Numa conjuntura tecnológica altamente ramificada, que evolui a cada minuto, há a necessidade de entender e mapear os fundamentos das linguagens de programação de alto nível e destapar as múltiplas máscaras evolutivas, em que encontram imersas, impedindo o entendimento do seu progresso. Da literatura electrónica, que tem vindo a emergir, surge a necessidade de mostrar e entender o seu desenvolvimento desde os seus primórdios. Para isso há a necessidade de definir literatura electrónica e focalizá-la para as obras de literatura gerada por computador de Pedro Barbosa que comportam o segmento artístico da poesia experimental e da geração combinatória textual.

Dessa forma, no Capítulo *Humanidades na Computação* são traçadas linhas temporais que descrevem o envolvimento das humanidades perante o advento da computação, assim como as linguagens de programação que foram usadas por Pedro Barbosa para a criação literária por computador. Este capítulo fornece contexto teórico necessário para que o leitor possa estudar o universo de conhecimentos que está na base do Capítulo *Especificação da Aplicação Web*.

O problema que a dissertação visa resolver está interligado à inexistência de uma edição crítica digital, que aglomere os trabalhos de Pedro Barbosa, com o devido cuidado na organização e selecção de obras de que a curadoria digital diz respeito e perante uma estrutura pensada para ser posteriormente replicada para outros autores de literatura electrónica.

Como solução para o problema, há a tentativa, no Capítulo *Especificação da Aplicação Web*, de reunir um conjunto de tecnologias que visem corresponder à definição do problema e, com isto, chegar a uma solução que sirva de contributo tecnológico no âmbito das Humanidades.

Capítulo 2

Humanidades na Computação

2.1 Percurso Cronológico das Humanidades na Computação

Esta secção traça uma linha temporal que visa proporcionar um entendimento relativamente ao envolvimento de investigadores e estudantes das Humanidades perante o advento da Computação no séc. XX.

2.1.1 Introdução

Antes dos feitos marcantes de Roberto Busa e da existência da IBM, a filosofia foi a primeira grande área das Humanidades a ter presença na computação por estar ligada à área da lógica, visto que "a Computação é filosofia — mais especificamente, o ramo da filosofia que se debruça sobre a lógica" (Ess, 2007), implicando que os computadores dependem de um sistema lógico para funcionarem. Tais sistemas remontam aos anos 40 e 50 do séc. XIX, arquitectados pelas mãos de George Boole. As primeiras experimentações na Computação por parte de filósofos estiveram adjacentes à lógica com o intuito de automatizar o ensinamento da lógica e explorar o computador como uma ferramenta para cálculo lógico.

Deste modo, surge a importância de mapear historicamente o envolvimento progressivo das Humanidades na Computação para entendermos o nosso presente. Que tecnologias e que conjunturas sócio-culturais estiveram no advento da evolução da Computação e em que termos se entrelaçaram com as Humanidades? Para responder a estas questões foram estabelecidos intervalos temporais que visam representar fases do paradigma computacional, demonstrando como é que as Humanidades ganham, gradualmente, a consciencialização das potencialidades da máquina para a resolução célere de problemas.

2.1.2 Os Primórdios da Computação

> "Ao contrário de múltiplas experimentações interdisciplinares, a Computação nas Humanidades teceu o seu início em 1949" (Hockey, 2004).

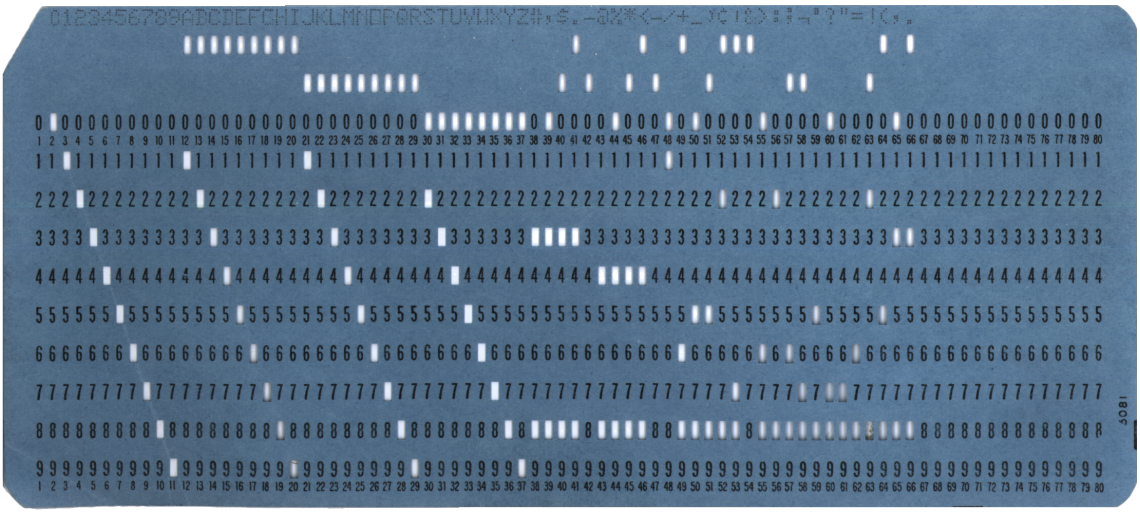


Figura 2.1: Cartões Perfurados

O padre italiano jesuíta Roberto Busa foi o primeiro a usar o computador para o processamento de palavras e textos, "após ter concebido a ideia de automatizar a análise linguística, entre 1942 e 1945" (Maio et al., 2006). Findada a tumultuosa Segunda Guerra Mundial, Busa construiu um índex de todas as palavras das obras de Santo Tomás de Aquino e de autores relacionados — num total de onze milhões de palavras em latim medieval. Para isso, Busa teve que criar métodos e terminologias para resolver o seu problema e os textos nas bibliotecas de nada lhe serviam — as suas ideias eram de natureza vanguardista para a época de então. No entanto, o arquitecto desta obra de dimensões excepcionais, cedo percebeu que uma máquina o poderia auxiliar a cumprir os objectivos. Sabendo da existência dos computadores, Roberto Busa procurou Thomas J. Watson da IBM para solicitar apoio na criação do intitulado *Index Thomisticus*. A obra de Busa teve luz verde perante a assistência pedida, e "[o]s textos foram transferidos para cartões perfurados (ver Figura 2.1) e um programa de concordância, e de análise textual, foi programado" (Hockey, 2004). Antes da existência do *HyperCard* (ver Figura 2.2) nos computadores da Apple Macintosh que permitia o uso do hipertexto, Roberto Busa tinha adicionado 300 códigos distintos, para cada uma das 11 milhões de palavras, — codificados em 130 bits contendo informação relacionada com a morfologia, fazendo dele "o primeiro ser humano a usar o hipertexto" (Maio et al., 2006).

Um programa de concordância e análise textual em que as palavras são alfabetizadas de acordo com as suas formas gráficas poderia ter dado azo a um resultado semelhante em menor tempo, mas Roberto jamais ficaria satisfeito com o seu produto final. O padre necessitava de produzir uma concordância lematizada[1] em que as palavras são listadas de acordo com os seus títulos no dicionário. Eventualmente, a equipa de Busa conseguiu escrever um algoritmo que permitia aplicar a lematização às 11 milhões de palavras e assim concluiu o seu índex, em 1949, num modo semi-automático, lidando os humanos com as formas das palavras que o computador não conseguia identificar. Mais tarde, em 1998, Busa palestrou na Hungria em relação às potencialidades da

[1] Lematização: consiste no processo de agrupamento de palavras similares como variações de um verbo ou substantivo — resultando num só termo para facilitar a pesquisa.

Figura 2.2: HyperCard da Apple

World Wide Web na difusão de materiais multimédia acompanhados de ferramentas sofisticadas de análise textual. Roberto Busa continua a ser um importante líder de opinião pela sua visão e imaginação, mesmo perante os praticantes do presente que nasceram com a era da Internet (Hockey, 2004).

Nos anos 60, outros investigadores começaram a vislumbrar os benefícios de usar concordâncias e escreveram uma série de artigos que faziam alusão a esta questão. Dolores Burton escreveu quatro artigos científicos para o jornal *Computers and the Humanities* frisando a importância de fundir os estudos das Humanidades com os computacionais (Hockey, 2004).

Apesar de haver um particular enfoque na produção de concordâncias, começava a emergir a consciencialização de que o computador possibilitava a implementação de métodos quantitativos. Outrora, em 1851, Augustus De Morgan, propunha, numa carta, um estudo quantitativo ao vocabulário com o intuito de investigar a autoria das *Epístolas Paulinas*. Mais tarde, T.C. Mendenhall, no final do séc. XIX, deu seguimento à ideia de Morgan ao arquitectar uma máquina de contar que era operada por duas mulheres que inseriam o número de palavras de duas, três, ou mais letras das obras de autores como William Shakespeare, Francis Bacon, Christopher Marlowe, entre outros, com o objectivo de determinar quem é que escreveu as obras de Shakespeare. A proposta de Augustus de Morgan vê os seus dias de vida quando, em 1963, Andrew Morton publicou um artigo afirmando que tinha, com o auxílio de um computador, concluído que o apóstolo Paulo só escreveu quatro das treze epístolas do *Novo Testamento* (Hockey, 2004).

O começo da era computacional torna, entretanto, a máquina de Mendenhall obsoleta. O computador permitiria armazenar frequências de palavras em quantidades superiores e com uma eficácia superior à de qualquer ser humano. Paradoxalmente, noutras áreas das Humanidades, como é exemplo da História, a era computacional afirmou-se com grande pujança.

Com o aproximar do final da década de 60, os historiadores começaram a sentir que estavam prestes a romper os cânones culturais da profissão, das práticas e da história em si mesma. Organizaram-se conferências que despoletaram discussões acesas entre reaccionários e inovadores. Os historiadores vanguardistas invocavam um ponto de vista alicerçado na teoria sócio-científica e na metodologia — contra os reacionários que reiteravam as práticas antigas: "Os «novos» historiadores usavam computadores para cálculos e conexões nunca antes executadas e os seus resultados eram, por vezes, excepcionais" (Thomas, 2004). O sucesso que acompanhava os novos historiadores era demasiado avassalador para ser derrubado pelos reaccionários. Os computadores e suas técnicas tornaram possível, no curso dos anos, alterar a percepção de como os historiadores encaravam as suas práticas. Emergia um paradoxo entre os que possuíam uma mente aberta e os que estavam dispostos a questionar esta nova cultura. Uns colocavam em prática ideias que iam desenvolvendo, outros mantinham-se inertes e cépticos em relação aos avanços que estavam a moldar as práticas no seio dos historiadores (Thomas, 2004).

A consciencialização da existência do paradigma computacional para os historiadores reacionários permanecia imperceptível. Um dos mais importantes investigadores da história nos EUA, Arthur Schlesinger, argumentava que a maioria das respostas relevantes a serem desvendadas eram susceptíveis de métodos qualitativos e nunca quantitativos. O grupo de Schlesinger questionava ainda se os alunos da nova geração necessitavam de projectos em grande escala e se os custos para contratar técnicos para os manter resultavam num investimento viável. Lawrence Stone, um historiador do Reino Unido de história moderna, chegou a referir que os orçamentos astronómicos que eram concedidos a investigadores em projectos altamente ambiciosos que visavam criar artefactos inovadores que combinavam a história com a Computação, no final ficaram aquém das expectativas (Thomas, 2004).

Segundo Robert Swierenga, os historiadores estiveram presentes na era computacional em 3 fases. Primeiramente, entre os anos 40 e 50, quando começaram a usar os cartões perfurados da IBM entre outras peças de *hardware* que lhes permitiam armazenar grandes quantidades de informação. Numa segunda fase, entre as décadas de 50 e 60, havia uma consolidação maior entre os historiadores e os computadores, despoletando tentativas de criação de *software* particularmente desenhado para ir ao encontro das necessidades dos historiadores. Na etapa final, a partir dos meados de 1960, "presenciava-se a existência de historiadores totalmente familiarizados com as linguagens de programação e a utilizarem um leque de *software* presente nas ciências sociais" (Swierenga, 1974).

Os anos 60 testemunharam ainda o aparecimento de espaços dedicados ao uso dos computadores nas Humanidades. No entanto, este novo paradigma ainda era percebido como sendo difícil e com uma complexa tecnologia associada, sendo essencialmente sinónimo de conjuntos de caracteres, de dispositivos de saída e entrada de código, e de sistemas de processamento de dados em

batch.

Apesar deste advento tecnológico ter sido um marco histórico na vida dos seres humanos, é importante reconhecer a existência de limitações notórias. A informação era constituída por números e textos que eram inseridos à mão ou por cartões perfurados — a título de exemplo, o trabalho em Fortran (ver Secção 2.2.2) de literatura gerada por computador de Pedro Barbosa contemplava este processo (ver Secção2.4.2). Cada cartão perfurado podia conter um máximo de 8 caracteres, ou uma só linha de texto. Todo o processamento de dados era em *batch*, significando que o utilizador só teria acesso aos resultados após a impressão da informação. A codificação de caracteres implicou um enorme problema na sua representação, o qual tem vindo a ser resolvido pela implementação do Unicode. Múltiplos métodos foram inventados para a representação de letras maiúsculas e minúsculas nos cartões perfurados, frequentemente pela inserção simples de um asterisco, ou caractere similar, precedendo uma maiúscula. No caso dos acentos, e outros caracteres que fugiam à norma, eram tratados similarmente e os alfabetos não romanos eram representados em transliteração (Hockey, 2004) .

Este novo paradigma viu emergir novas oportunidades para os interessados na Computação em Humanidades poderem partilhar ideias e problemas. Em 1964, a IBM organizou uma conferência em Yorktown Heights que fomentou, no ano seguinte, a publicação de *Literary Data Processing Conference Proceedings* editada por Jess Bessinger e Stephen Parrish. Os artigos discutiam questões relacionadas com a codificação do material do manuscrito e na organização automática das concordâncias — em que tanto as variantes ortográficas quanto a falta de lematização causavam sérios obstáculos.

Porventura, a ocorrência das conferências em Yorktown Heights era irregular e, como forma de contornar esse facto, deu-se início à primeira série regular de conferências em Computação literária e linguística que vieram mais tarde resultar nas conferências da *Association for Literary and Linguistic Computing/Association for Computers and the Humanities* (ALLC/ACH), organizadas por Michael Farrington e Roy Wisbey, fundador do *Centre for Literary and Linguistic Computing* da University of Cambridge, em Março de 1970. As actas dessas conferências, editadas em 1971 por Wisbey, facultaram informação que serviu para traçar uma linha editorial para as publicações subsequentes. O conteúdo das actas transparecia interesses relacionados com a entrada e saída de código na programação, assim como com a lexicografia, a edição textual, o ensino de línguas, e as estilísticas. Nesta altura, havia já a consciencialização da necessidade da existência de uma metodologia para o arquivo e manutenção de textos electrónicos (Hockey, 2004).

Esta fase veio consolidar os avanços anteriormente alcançados. Mais pessoas estavam a incluir as metodologias desenvolvidas até então e assim surgiu um acréscimo na quantidade de textos electrónicos e de projectos a serem produzidos. A difusão das potencialidades do computador despoletou, no seio académico, um acréscimo de utilizadores que exploravam as ferramentas computacionais com o intuito de melhorarem a qualidade dos seus trabalhos — aplicando-as à investigação e ao ensino. As conferências da *Computers and the Humanities* e de outras séries de colóquios independentes permitiram a célere transmissão de conhecimento na área das Humanidades. As séries bianuais de conferências tiveram início após o simpósio de 1970 em Cambridge, o

qual estimulou a produção de artigos de alta qualidade. A meio da década de 70, nasceram outras séries de conferências nos EUA, denominadas por *International Conference on Computing in the Humanities* (ICCH).

Perante a conjuntura vivida nos primórdios deste paradigma, surgiu a necessidade de criar *software* que evitasse a perda de tempo a escrever intermináveis linhas de código de programação e que, por sua vez, permitisse que o custo dos projectos fosse mais reduzido. Com esta procura, começaram a ser comercializados pacotes genéricos de *software* (por exemplo: WordCruncher) para facilitar a criação de textos electrónicos.

O lançamento do controverso livro *Time on the Cross: The Economics of American Negro Slavery* (1976), por (Fogel, 1995), em dois volumes, despoletou o início de múltiplos debates acerca do uso de métodos computacionais na área de investigação da história. O livro de Fogel e Engerman não só expunha dados referentes à escravatura e à sua rentabilidade económica, um dos temas mais acesos da história norte-americana, como era uma obra que conjugava metodologias quantitativas e análise histórica. Uns aplaudiam o sucesso dos autores, outros referiam que a obra colocou de parte questões importantes de análise quantitativa e textual de outros historiadores, além do contexto político e social da escravatura. Nisto, os críticos de *Time on the Cross* expressavam que o livro "estava demasiado concentrado nas interpretações de dados em análises quantitativas pelos autores e endereçava maioritariamente questões económicas" (Thomas, 2004).

Entretanto, começou a sentir-se a necessidade de armazenar os textos electrónicos que eram produzidos, de modo a evitar a perda permanente das obras — considerando a fragilidade da tecnologia de armazenamento físico de então. Em 1976 nasceu a iniciativa *Oxford Text Archive* (OTA) para assegurar que os textos electrónicos tinham a vida garantida. Após estarem finalizados, os textos eram incluídos numa base de dados pesquisável por investigadores para proporcionar a exibição e disponibilização, consoante os critérios de direitos de autor das obras. Com a OTA, assistimos aos primórdios da biblioteca digital (Hockey, 2004).

Sucedaneamente, o ensino viu nascer as primeiras unidades curriculares que visavam cobrir diversos aspectos da Computação nas Humanidades. Algumas destas unidades curriculares eram leccionadas por funcionários dos centros académicos de Computação — especificamente nas questões de uso de *software*. As cadeiras ensinadas por professores possuíam, tendencialmente, os interesses dos docentes — resultando na criação, por parte dos alunos, de projectos nas mesmas áreas. O dilema relacionado com a importância ou insignificância do ensino de linguagens de programação aos alunos estava entretanto a ser debatido. O argumento central a favor do ensino de código computacional reiterava que "oferecia uma abordagem mental correcta – substituía o Latim como proporcionador de disciplina mental" (Hockey, 2004). Certos autores argumentavam que o ensino do código era a única forma de os alunos estarem consciencializados acerca das potencialidades e limites do computador — despoletando uma abertura de mente nas Humanidades, abrindo novos caminhos de discussão e descobertas na área. Contudo, o ensino de programação a alunos de Humanidades requeria docentes proficientes que, por sua vez, possuíam maior credibilidade em detrimento dos programadores das Humanidades. Outros autores afirmavam que as linguagens de programação eram demasiado difíceis e o tempo que despendiam deveria ser usado para uni-

dades curriculares mais relevantes do seio das Humanidades. Nos anos 70, as linguagens Snobol, BASIC e Pascal estavam na moda e eram as mais atraentes para os estudantes, em detrimento da linguagem Fortran que era a mais usada.

A substituição da cassete pelo disco magnético (por exemplo: um dos leitores de discos era o IBM 1316 — ver Figura 2.3) no armazenamento de informação veio trazer avanços significativos nas ferramentas de processamento. A pesquisa de ficheiros nas bases de dados deixou de ser sequencial. Durante anos houve várias tecnologias para a organização de informação de uma forma eficaz na área das Humanidades, mas o modelo relacional de disposição das obras começou gradualmente a ter preponderância pela sua estrutura. Contudo, os modelos relacionais apresentavam problemas ao nível da representação da informação, que tinha que ser inserida em tabelas. Nos anos 70, foram inventadas duas peças de *hardware* que auxiliavam a pesquisa: uma implementada no computador *Ibycus* de David Packard, e uma outra denominada *Content Adressing File Store* que era compatível com os computadores ingleses da *International Computers Limited* (ICL). O conceito de transmissão de processamento para *hardware* era muito bem visto pelos investigadores de Humanidades que lidavam com avultadas quantidades de material. Apesar destas invenções, o *hardware* convencional evoluiu na sua velocidade de processamento — deixando a competição para trás (Hockey, 2004).

2.1.3 Integração das Humanidades na Computação

Com o nascimento do computador pessoal e do correio electrónico, assistimos a uma maturação do paradigma das Humanidades na Computação. No que toca a outras tecnologias, algumas caíram na obsolescência, outras progrediram mediante as necessidades dos seus utilizadores.

Verifica-se igualmente o surgimento de marcas, para preencher e competir no novo mercado. Algumas empresas criavam computadores para os jogadores de vídeo jogos, outras produziam computadores unicamente para processamento de texto. A IBM foi a marca que se destacou, inicialmente, na produção de portáteis e de sistemas operativos que foram posteriormente adoptados noutras empresas de *hardware*. A Apple Macintosh surge em segundo lugar, retendo um segmento de utilizadores atraídos pela sua interface gráfica, em contraste com a IBM ou a Microsoft.

A Apple Macintosh primava por ser a única marca com uma interface gráfica que por sua vez possibilitava a mostra e manipulação de caracteres não convencionais que podem ser encontrados nos alfabetos do Inglês tradicional, Grego, Cirílico, e afins. Outra vantagem advém da possibilidade de criação de hipertexto através do *HyperCard* que consiste num modelo em pilha de cartões virtuais que armazenam informação e que, por sua vez, estão interligados entre si — permitindo interactividade e a existência do hipertexto. Os modelos criados em *HyperCard* podiam ser transferidos para uma disquete para posteriormente serem fisicamente partilhados e digitalmente lidos em computadores. A título de exemplo, Patrick Conner, em 1991, criou a *Beowulf Workstation* — "uma pilha de cartões virtuais *HyperCard* com vista a auxiliar os alunos a prepararem a tradução e explicação de um poema em Inglês antigo" (Conner, 1991). Os alunos de Conner tinham à disposição um conjunto de hiperligações nos cartões virtuais para textos em Inglês moderno com anotações linguísticas e contextuais.

Figura 2.3: IBM 1316: leitor e armazenador de discos magnéticos

O computador pessoal torna-se um bem essencial para os académicos que conheciam o modo de operar da máquina e as suas possibilidades. Em consequência, os estudantes de Humanidades detentores de portáteis já não necessitavam dos centros de computadores para levar a cabo a sua investigação.

Estávamos nos finais da década de 80 quando surgiram três *software* de análise textual baseados em DOS: o Text Analysis Computing Tools[2] (TACT), o WordCruncher[3], e o Micro-OCP[4]. Os utilizadores do WordCruncher e TACT obtinham resultados imediatos nas suas procuras pelos seus modos de pesquisa interactiva ao contrário do Micro-OCP, cujo sistema de pesquisa consistia numa *mainframe* que usava uma técnica de concordância em *batch*.

À medida que o paradigma da Computação nas Humanidades cria ramificações e evolui, assistimos ao início da implementação do correio electrónico como serviço público com origens na conferência de ALLC[5] em 1985 em Nice, onde *emails* eram trocados frequentemente. Anteriormente, este sistema era somente usado por investigadores e cientistas em Computação.

Passados dois anos, teve lugar a conferência ICCH na cidade de Columbia na Carolina do Sul. Aí, um grupo de académicos na área da Computação nas Humanidades reuniu-se e deliberou que deveria encontrar uma forma de trocar informação, diariamente, entre si. Dos elementos do grupo, William McCarty descobriu *ListServ*, um *software* que funcionava como um gestor automático de contactos de correio electrónico, desenvolvido em 1986 por Eric Thomas. Por consequência, *ListServ*, permitia enviar *emails* que eram difundidos para todos os endereços de correio electrónico presentes numa dada base de dados. Com isto, McCarty decide criar a *Humanist*, uma conferência electrónica internacional sobre a Computação e o digital nas Humanidades. Como explica (Hockey, 2004), "a *Humanist* foi central para a manutenção e desenvolvimento de uma comunidade, proporcionando uma contribuição significante na definição da Computação nas Humanidades".

Outra das inovações que abalou este período da década de 80 foi introduzida por Nancy Ide em colaboração com os seus colegas, a Novembro de 1987. Ide organizou uma reunião no colégio de Vassar, em Poughkeepsie, para discutir a possibilidade de criar um esquema de codificação padronizado com vista a contornar o obstáculo permanente que estava adjacente aos diferentes tipos de codificação presentes em diferentes *software* de edição de texto. O tempo que era despendido na reformatação de textos para serem lidos por um determinado *software* despoletava um problema que tinha que ser endereçado. Embora, no ano anterior, em 1986, tivesse surgido um esquema de codificação criado pela *Standard Generalized Markup Language* (SGML) e publicado pela *International Organization for Standartization* (ISO), este novo método oferecia um sistema para

[2]Text Analysis Computing Tools (TACT) é um sistema de análise e retorno textual em MS-DOS de pedidos por parte do utilizado a bases de dados em linguagens europeias. O TACT fora desenvolvido entre 1986 e 1989 na *IBM-University of Toronto Cooperative in the Humanities* por uma equipa composta por John Bradley, Ian Lancashire, Lidio Presutti e Michael Stairs — `http://projects.chass.utoronto.ca/tact/`

[3]WordCruncher é um *software open-source* para a leitura de eBooks que contempla ferramentas de pesquisa para auxiliarem investigadores e estudantes: `http://www.wordcruncher.com/`

[4]Micro-OCP ou Oxford Concordance Program é uma ferramenta para gerar concordâncias, listas de palavras e indexes de textos de qualquer língua ou alfabeto — `http://users.ox.ac.uk/~ctitext2/resguide/resources/o125.html`.

[5]ALLC ou Association for Literary and Linguistic Computing foi criada em 1973 para auxiliar o campo da Computação no estudo da literatura e linguagem (Eden, 2005).

a definição de uma estrutura de codificação que conseguisse ler diferentes tipos de texto, dados e meta-dados. A SGML entrou em desuso por ser demasiado complexa; mas é vista com uma precursora de HTML (Hockey, 2004).

Transversalmente, a conferência organizada por Nancy viu os seus membros assinarem um conjunto de princípios denominados por *The Poughkeepsie Principles* como "base para a construção de um esquema novo de codificação, confiando a gestão do projecto ao *Steering Committee* com representantes da ACH[6], ALLC, e da *Association for Computer Linguistics*" (Hockey, 2004). Esta iniciativa resultou numa angariação de mais de 1 milhão de dólares nos EUA, proporcionando o nascimento da *Text Encoding Initiative* (TEI): *Guidelines for Electronic Text Encoding and Interchange*. A TEI[7] operava em quatro áreas, contendo cada uma delas um comité que fazia chegar toda a informação produzida a dois editores que posteriormente tratavam da disseminação das ideias, iniciativas e inovações da TEI. Em 1994 saiu a primeira publicação da *TEI Guidelines* em formato papel e digital (Hockey, 2004).

O reconhecimento da importância da TEI no seio da Computação nas Humanidades foi instantâneo por ser a primeira tentativa de categorizar e definir todas as particularidades contidas nos textos em Humanidades. Com isto, foram especificadas 400 *tags* de codificação numa estrutura extensível a uma panóplia de áreas de aplicação.

Estes progressivos avanços na Computação e na sua relação com as Humanidades é concomitante com o nascimento dos computadores pessoais, do email, do trabalho em rede e da TEI, que continuaram a progredir e a ver um acréscimo em projectos e utilizadores. No entanto, a Computação linguística começou, gradualmente, a distanciar-se da Computação nas Humanidades. A perda de aspectos linguísticos da Computação nas Humanidades foi substancial — principalmente no corpus linguístico. A organização de conferências e reuniões passou a ser ocasional. Factualmente, ambas as áreas cresceram sempre separadamente mas houve tentativas de as unificar. A título de exemplo, Don Walker da *TEI Steering Committee* e Antonio Zampolli do *Institute for Computional Linguistics* tentaram cruzar as duas áreas, mas esse enlace apenas foi sentido nos primórdios da Computação nas Humanidades, altura em que os académicos necessitavam de ferramentas que eram desenvolvidas na linguística da Computação, nomeadamente: análise morfológica, análise sintática ou bases de dados lexicais (Hockey, 2004).

2.1.4 A World Wide Web

A década de 90 revolucionou os parâmetros de comunicação pela introdução da World Wide Web, o rosto visível da Internet, por Tim Berners-Lee. Em 1993, surgira *Mosaic*[8] (ver Figura 2.4),

[6]ACH ou Association for Computers and the Humanities é uma grande organização internacional fundada para pessoas de investigam sobre ferramentas computacionais para as Humanidades — específicas para a manipulação de materiais textuais nas áreas de filosofia, história, literatura e estudos linguísticos (Eden, 2005).

[7]TEI ou Text Encoding Initiative foi um dos primeiros padrões de meta-dados criado para facilitar a codificação de textos literários para mostra e pesquisa na Internet (Eden, 2005).

[8]Mosaic, ou NCSA Mosaic, é um *browser* hipermédia para a WWW desenvolvido na National Center for Supercomputing Applications (NCSA) na Universidade de Illinois, e disseminado em 1993. O Mosaic permitia a visualização de texto, hipertexto, imagens, áudio e vídeos; para além destas características, comportava ferramentas de acesso como a gopher e UseNet (Morris, 1995).

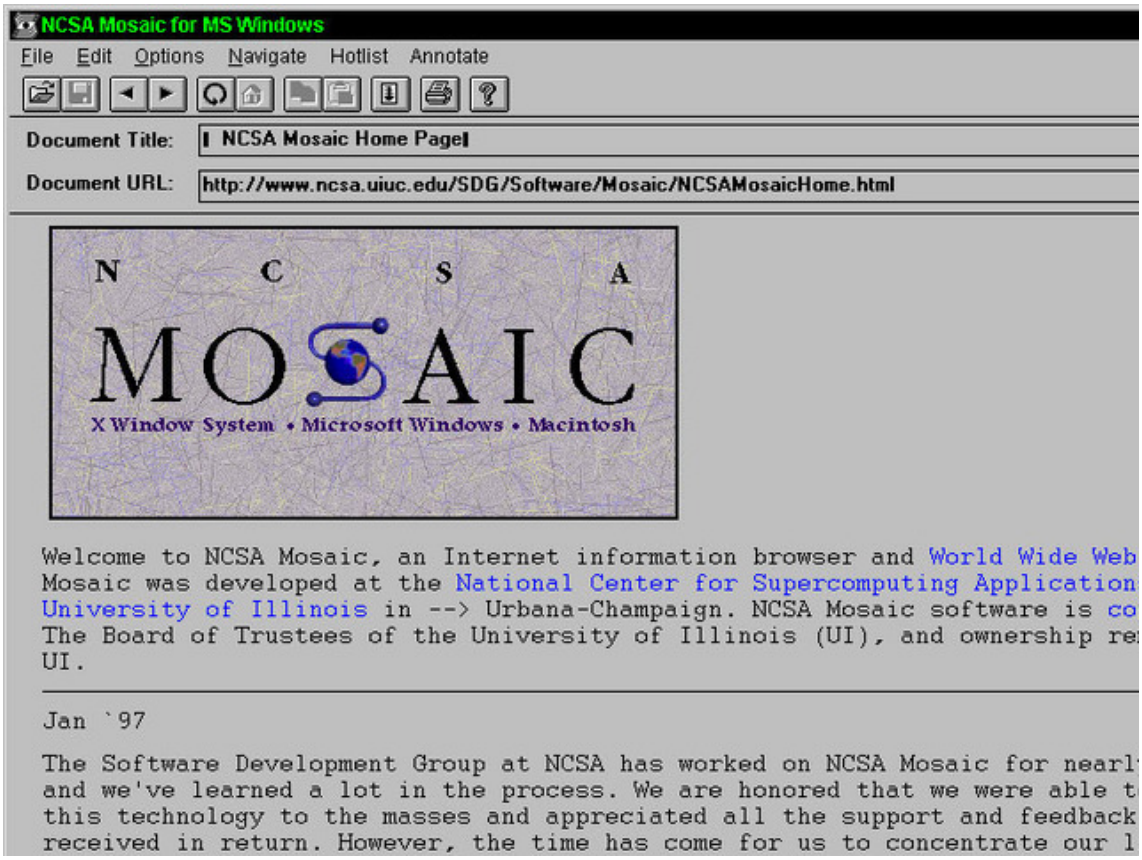


Figura 2.4: Mosaic: o primeiro browser para a WWW

o primeiro *browser*. O impacto do nascimento deste novo paradigma comunicacional foi imediato nos investigadores de Computação nas Humanidades. Na TEI fluía o sentimento de desvalorização perante a linguagem em *tags*: *HyperText Markup Language* (HTML) por ser "um sistema de marcação fraco que perpetuava todos os problemas com processadores de texto" (Hockey, 2004). Assim, a Web era vista como um meio para encontrar alguns tipos de informação mas não como uma ferramenta credível de investigação em Humanidades. Transversalmente, a Internet despoletava a oportunidade para as organizações e instituições que idealizavam, futuramente, iniciar um envolvimento na Computação nas Humanidades.

A democratização da difusão de informação cresce com a Web, já que qualquer pessoa com conhecimentos de HTML e linguagens relacionadas pode publicar abertamente para toda a comunidade de cibernautas sem limitações causadas por outrem (a título de exemplo: editoras, chefes de redacção, etc). As restrições no formato e número de caracteres são inexistentes e o hipertexto assume novos contornos. Contrariamente aos artefactos de formato físico, como no caso de livros, jornais ou revistas, as páginas Web são obras abertas: possibilitam uma interminável adição e edição de informação.

Mediante a revolução comunicacional que a World Wide Web causou, escritores e investigadores organizaram conferências, publicaram artigos e iniciaram angariações de fundos para projectos. De acordo com (Hockey, 2004), "Peter Robinson e poucos mais viram os seus projectos

passar da teoria à prática, porventura daí só surgiram protótipos ou pequenas demonstrações do que os projectos se poderiam tornar quando finalizados". Uma das grandes dificuldades de colocar em prática os projectos idealizados residia no desenvolvimento de *software* e na marcação do texto — resultando num enorme consumo de tempo.

O debate no seio da comunidade de estudos da relação entre Computação e Humanidades persistiu, propiciando o nascimento de novas ideias e a partilha de conhecimento. Com o início do paradigma da Internet, assomavam-se panóplias de recursos electrónicos denominadas por "arquivos" — termo aceite pela vasta maioria dos investigadores para designar as colecções em formato electrónico. A título de exemplo, em 1996 fora criado o *The William Blake Archive* que contemplava uma colecção de livros, manuscritos, pinturas e imagens das obras de arte do artista para furtuito de académicos e investigadores com acesso à Internet situados em qualquer ponto geográfico do planeta (Jones, 2006). Para (Hockey, 2004) "o arquivo significava uma colecção de material onde o utilizador teria que normalmente escolher uma rota de navegação". A TEI continuava a dar preferência a SGML como linguagem de marcação de textos para a estrutura em que as rotas de navegação podiam ser construídas. No entanto, a SGML era demasiado limitada no desenho e criação de uma interface eficiente em contraste com o HTML.

Não obstante a grande generalidade dos primeiros projectos serem publicados na Internet por investigadores, também os bibliotecários rapidamente consideraram a colocação de conteúdo das suas colecções na Web. Nos EUA, uma multitude de bibliotecas disponibilizou textos electrónicos ou colecções digitais para a área das Humanidades que poderiam ser encontrados através do motor de busca OpenText SGML.

A possibilidade de produção colaborativa em projectos através da Internet é revolucionária: pessoas podiam contribuir para o mesmo projecto sem estarem espacialmente condicionadas. Posteriormente, surgiu a possibilidade de pessoas adicionarem anotações aos documentos, como é o caso do *The Peirce Edition Project*[9]. Paralelamente, a multimédia acrescentava informação em formato de imagens, som e vídeo — proporcionando uma experiência multimodal mais rica. A questão da modularidade da informação surgia pouco tempo depois, especialmente na imagem, através de *software* ou algoritmos que permitiam a manipulação dos dados. "A expansão do acesso a recursos electrónicos fomentados pela Web levou a outras áreas de interesse teórico na Computação nas Humanidades" (Hockey, 2004).

Na viragem para o séc. XXI, Lev Manovich publica *The Language of New Media* pela MIT Press, obra conotada como a mais relevante desde *Understanding New Media* por Marshall McLuhan. Manovich, no seu livro, introduz o novo paradigma dos meios de comunicação digitais, contrastando-o com o tradicional. O autor parte de 5 princípios que estão intrinsecamente presentes nos novos média: *representação numérica*, *modularidade*, *automação*, *variabilidade* e *transcodificação*. De acordo com (Manovich, 2001), "nem todos os objectos dos novos média obedecem a estes princípios; não devem ser considerados como leis absolutas mas como tendências gerais de uma cultura que está passar pela computorização".

[9]The Peirce Edition Project: `http://www.iupui.edu/~peirce/`

O princípio da *representação numérica* diz respeito ao facto inalienável de que todos os objectos nascidos no meio digital, ou convertidos do analógico para o digital, possuem uma representação binária que, consequentemente, pertence a um conjunto de algoritmos passíveis de manipulação. Os novos média tornam-se programáveis; num gráfico SVG (linguagem XML) é possível substituir uma cor ao reprogramar os campos da instrução *fill* no código de programação que está ocultado e que tem o papel de gerar uma representação, neste caso gráfica, do objecto.

Na geração do computador, o conceito de *modularidade* assume um papel preponderante na estruturação dos novos objectos mediáticos. Todos os objectos se tornam passíveis de serem fragmentados, sem perderem a sua identidade, qualidade ou informação, podendo ser recombinados num número infinito de versões. Ao nos apropriarmos de um objecto mediático digital, podemos recombiná-lo com outros tantos para, no final, termos um artefacto "novo" com objectos incorporados de múltiplos autores (Manovich, 2001).

Com isto, Manovich introduz a *automação*, terceiro princípio que está directamente ligado aos anteriores. Neste conceito, o humano somente está "envolvido na criação, manipulação e acesso dos média" (Manovich, 2001) já que a *automação* nos novos média está na natureza do computador e na sua capacidade de executar operações. Há um leque de *software* que permite a automação de tarefas como a edição de imagem ou áudio, processamento de texto, criação de gráficos 3D. Estes tipos de *software* comportam funcionalidades que permitem auxiliar o utilizador na sua criação para obter o resultado melhor possível. A título de exemplo: editores de texto para programar com *autocomplete* para aumentarem a eficácia e eficiência de escrever código.

Sequencialmente, a *variabilidade* advém da fusão dos conceitos de *representação numérica* e *modularidade*. Nos média tradicionais, uma criação continha uma ordem linear determinada pelo seu criador, sendo posteriormente armazenada num formato físico para ser exibida ou copiada. Em oposição, os novos média abolem a linearidade, possibilitando a recriação de uma obra num número infinito de versões. Com explica (Manovich, 2001), "[o]s elementos estão fragmentados em amostras discretas (ex: uma imagem é representada por uma cadeia de pixeis), podem ser criados e personalizados na hora". Nesta linha de raciocínio entra a imediatez, a capacidade de um utilizador aceder a uma multitude de informação armazenada em bases de dados espalhadas pela Internet, seleccionar os conteúdos, atribuir-lhes uma sequência e finalmente dar ordem ao computador para gerar a obra. O utilizador assume duas facetas: consumidor e produtor de conteúdo; apropria-se do conteúdo de outrem para o recombinar e produzir uma versão de sua autoria, sem descartar os autores do conteúdo que fora apropriado, que poderá ou não possuir informação original.

No último princípio de Manovich, *transcodificação*, o autor entende que há a existência de duas camadas distintas mas continuamente interligadas: *cultural* (ex: composição e interpretação; história e narrativa) e *computacional* (exemplos: selecção e combinação; linguagem de programação e estrutura de dados). Lev Manovich justifica a existência destas duas camadas com base nos seguintes argumentos: "os novos média são criados com o computador, distribuídos pelo computador e arquivados no computador" (Manovich, 2001). Assim, entendemos que a *camada cultural* representa as acções tradicionais em que a cultura humana preenche o mundo, e por outro

lado os meios, *software* e *hardware* que o computador, na *camada computacional*, possuem para representar o mundo.

Entre a *camada cultural* e a *camada computacional*, Stephen Ramsay estabelece uma ponte rígida entre o estudo dos média com o estudo computacional. Ramsay denomina este conceito por *crítica algorítmica*, exemplificando que a palavra 'crítica' deriva da manipulação algorítmica de texto. O autor defende que a revolução digital está ainda por penetrar o núcleo central dos estudos literários, apesar das "inúmeras revoluções no campo epistemológico, esta continua mais preocupada na análise interpretativa dos artefactos culturais escritos" (Ramsay, 2011). Para uma *crítica algorítmica* emergir, teria que haver uma alteração filosófica. Os textos são procurados, investigados e difundidos por todos — raras são as ocasiões em que são transformados em algoritmos com resultado a entrarem num espaço de interpretação crítica.

À medida que os artefactos digitais em Humanidades acrescem em complexidade para tirar partido das capacidades da Web, há cada vez mais uma aproximação dos estudos computacionais, de *software*, que requerem um cuidado especial, uma indagação própria e uma inclusão obrigatória no estudo dos média. David Berry reforça a necessidade de examinar o campo complexo da cultura vista pela tecnologia digital. Berry defende que para compreendermos a cultura contemporânea dos artefactos nascidos em ambiente digital teremos que nos focar na programação computacional que está intrínseca em todos os aspectos de cultura e de memória — incluindo a reflexão referente à presença do código. Segundo (Fuller, 2008), "de alguma forma, todo o trabalho científico/intelectual é agora um estudo de *software*, na medida em que o software providencia o seu *medium* e o seu contexto".

Em pleno ano 2015, nunca antes foi tão acessível para um investigador ou estudante de Humanidades conseguir escrever e perceber código, afirmando-se no mundo da comunicação digital. Há um contínuo crescendo de linguagens e plataformas de programação de alto nível que viabilizam a fácil criação de artefactos nascidos em meio digital. Perante as linguagens Web que são mais susceptíveis de serem usadas pelas Humanidades, como são exemplo: HTML5, CSS3, JavaScript, JQuery, PHP, SQL, entre outras — há plataformas que facilitam a sintaxe de algumas linguagens, como é o caso do Laravel[10] ou do CakePHP[11] que simplificam o uso de PHP. O Sass[12] e LESS[13] expandem as funcionalidades de CSS3 e reduzem linhas de código. Nos editores de texto de código vêm incorporadas funcionalidades de completação automática presentes no SublimeText3 ou Brackets. Os *Content Manager Systems* (CMS) facilitam o uso de *back-end* ao possuírem uma interface que alicerça o utilizador com múltiplas funcionalidades e módulos como é o caso do WordPress ou Joomla. Os sistemas de depuração de erros incorporados em plataformas

[10] Laravel PHP Framework: `http://laravel.com/`.

[11] CakePHP: `http://cakephp.org/`.

[12] Sass ou *Syntactically Awesome Style Sheets* (`http://sass-lang.com/`) é um pré-processador, programa que opera no lado do cliente, que oferece extensões e estenografia numa linguagem equivalente ao CSS3, que é posteriormente transformada em CSS3 através de compiladores com Compass (`http://compass-style.org/`) e Gulp (`http://gulpjs.com/`). Estas extensões contemplam variáveis e regras para concentrar selecções em selecções gerais, e funções personalizáveis que dão mais liberdade ao utilizador — algo que o CSS puro não possibilita (Gasston, 2013).

[13] LESS é equivalente a Sass.

(ex: Susy para SASS) ou em *browsers* (por exemplo: Firebug aditivo do Firefox) auxiliam a célebre identificação de incongruências no código. As plataformas Web tais como Twitter Bootstrap, Foundation[14] ou Font Awesome[15] comportam componentes Web, facilitando a criação de objectos e aplicações Web. A existência de *software* que permite a criação de plataformas digitais sem o utilizador escrever uma linha de código despoletam uma grande atractividade como é o caso de Adobe Muse [16] ou Microsoft Project Siena[17]. Perante a parafernália de ferramentas ao nosso dispor, a maioria das quais gratuitas, as possibilidades de criação de artefactos para as Humanidades são quase infinitas.

Estamos perante uma conjuntura cultural de dimensão computacional. A importância inegável de que temos que perceber e engajar com o código. "O código computacional funciona como um índex da cultura digital" (Ramsay, 2011).

2.1.5 Curadoria Digital

Antes de se diluir no seio digital, o conceito de curadoria e a actividade de um curador estiveram sempre presentes em departamentos, museus, bibliotecas que têm vindo continuamente a arquivar uma multitude de géneros de artefactos físicos; desde livros que emanam a aura do seu autor, a fotografias, manuscritos, gravações e afins — que conservam informação de especial relevância para curadores, investigadores e público em geral. A humanidade, a partir do séc. XX mas mais enfaticamente após a sua segunda metade, assistiu a um progresso tecnológico célere comparativamente aos séculos anteriores; deste rumo evolutivo "surtiram tecnologias digitais e analógicas em novos formatos, dispositivos e métodos de produção" (Sabharwal, 2015).

O metadado descritivo[18] foi uma das tecnologias adoptadas desde cedo para aumentar a acessibilidade e facilidade na pesquisa, assim como para o controlo de procedimentos de autenticação — permitindo criar mecanismos de auditoria para assegurar que o material não seria acedido ou alterado pelos desautorizados para tal (Higgins, 2011).

A consciencialização para a necessidade de preservar artefactos digitais fora iniciada nos EUA em 1996 quando a US Task Force on Archiving of Digital Information escreveu um relatório que indicava a necessidade do desenvolvimento de estratégias que garantissem a preservação e resultante sobrevivência de artefactos digitais culturalmente valiosos. O conjunto de objectivos visados no relatório enfatizava a importância das organizações levarem a cabo actividades de preservação digital e de criarem cargos para integrarem capital humano nessas tarefas. As funções desses cargos traduziam-se na gestão dos arquivos e na migração de artefactos de modo a os salvaguardar da obsolescência. O relatório serviu de ponto de partida para a curadoria digital e despoletou uma

[14] Foundation: `http://foundation.zurb.com/`.
[15] Font Awesome: `http://fortawesome.github.io/Font-Awesome/`.
[16] Adobe Muse: `http://www.adobe.com/pt/products/muse.html`.
[17] Microsoft Project Siena: `https://www.microsoft.com/en-us/projectsiena/`.
[18] O Metadado descreve um recurso para fins de pesquisa ou identificação; poderá incluir elementos como o título, resumo, nome de autor e palavras-chave.

iniciativa internacional que resultou na criação de metodologias e ferramentas (Data Asset Framework (DAF)[19] e DRAMBORA[20] e conferências internacionais para fomentar a troca de ideias e discutir progressos (Higgins, 2011).

A curadoria digital e as Humanidades digitais operam de modo a preservar, promover e garantir o acesso a colecções digitais. As suas ferramentas comportam uma arquitectura de informação, média sociais, gestão de conhecimento que contribuem significativamente para as três componentes que residem na ferramenta: interacções sociais, prática e teoria. Deste modo, há um ecossistema para a curadoria digital que se rege por conteúdo Web que é catalogado, de referências cruzadas, com pontuação, com filtração de resultados, e gerido por seres humanos. Consoante o crescendo do paradigma digital, os curadores e arquivistas assistiam à criação de obras nascidas no ventre do digital que requeriam novos e diferenciados métodos de arquivamento e preservação; conferências foram organizadas com bibliotecários, arquivistas e programadores no início do séc. XXI para endereçar os requisitos do meio digital. A curadoria digital proporciona um espaço para a contínua preservação de artefactos digitalizados ou nascidos digitalmente a investigadores nas Humanidades e outras áreas de estudo, a comunidades de acesso livre (a título exemplo: Wikipédia), a instituições, a professores, entre outros, que indagam por meios que permitam integrar recursos digitais quer sejam do foro académico, pessoal ou profissional (Sabharwal, 2015). Os investigadores nas Humanidades digitais têm vindo a aplicar exaustivamente a Computação à investigação. A necessidade de aceder livremente e facilmente a recursos a dados primários (por exemplo: recolhas de informação por entrevistas a pessoas-chave num determinado assunto) e a toda uma herança cultural "implica dicotomicamente o contínuo aumento de conhecimento nas Humanidades e, por sua vez, o uso de novas metodologias — impossíveis sem a tecnologia digital" (Sabharwal, 2015).

2.1.6 Obsolescência Tecnológica

Correntemente vivemos numa sociedade de contínua obsolescência tecnológica — em consequência, os materiais electrónicos vão enchendo lixeiras espalhadas pelos cantos do Mundo. Associamos o termo "obsolescência" à morte tangível ou intangível de algo. Nos EUA, em 2004, "cerca de 315 milhões de PCs em bom funcionamento foram removidos e somente 10% foram actualizados" (Jackson, 2008). A obsolescência tecnológica do *hardware* está intrinsecamente ligada à do *software* —, este será o foco principal desta subsecção.

Neste projecto documentamos linguagens de programação que foram relevantes para investigadores, programadores, entre outros, num dado espaço temporal para o desenvolvimento de *software*, para resolução de problemas e automatização de tarefas, além de servirem de alicerce para o nascimento de linguagens que comportam características que melhor se coadunam com as necessidades de uma civilização em constante movimento.

[19] A DAF desenvolveu um método para enumerar e auditar os detentores dos direitos de autor dos conteúdos (Higgins, 2011): http://www.data-audit.eu/users.html

[20] DRAMBORA ou Digital Repository Audit Method Based on Risk Assessment é uma ferramenta para analisar os potenciais riscos associados à sustentabilidade e continuidade de repositórios digitais: http://www.repositoryaudit.eu/

A premissa conservadora que sugere que os engenheiros e *designers* se devem preparar para a obsolescência é contra-argumentada por Rob Lawlor no seu artigo *Delaying Obsolescence* (Lawlor, 2015). Planear para a obsolescência pressupõe um conjunto de medidas e decisões que devem ser tomadas antes da concepção do produto para minimizar o seu impacto no ambiente, contemplando como é que o produto será usado — estimando o seu tempo de uso e em que contornos a destruição do produto no fim de vida será levada a cabo. Sugere um planeamento apropriado para a obsolescência para que os produtos estejam melhor preparados para a chegada do seu prazo de expiração. Como forma de atrasar o estado de obsolescência em que um produto tecnológico, perante o *status quo* que reside na mente do consumidor (exemplo: consumidores leais à Apple e com poder de compra, tiram imediato partido dos novos produtos — ficando para trás máquinas electrónicas em perfeito estado funcional), é de extrema relevância que o desenho do produto seja trabalhado no seu curso de existência com o intuito de se manter atraente e útil. Contudo, a actualização de *software* ou quiça de sistema operativo (SO), não obstante de estar associada a um reforço dos recursos do SO ou *software* resultando, maioritariamente, numa melhor experiência de utilização — implica, por sua vez, uma desadequação das mudanças ao sistema (a título de exemplo: um computador portátil que tenha por omissão o Windows 7, sofrerá acréscimos no tempo de resposta aquando da instalação de versões de SO como Windows 8, 8.1 ou 10 que consumem mais recursos de sistema). Como explica (Malinconico, 1984), "estas «actualizações» feitas num sistema automático, contrariamente a melhoramentos num loja física (ou *brick and mortar*[21]), não apreciam o seu valor; pelo contrário, reduzem o tempo de substituição de *software*".

A problemática da obsolescência tecnológica serve de especial reflexão e referência para contextualizarmos o percurso cronológico traçado na Secção 2.2 (Linguagens de Programação na Literatura Electrónica). A cultura de *software*, que evolui e se complexifica a cada segundo, assenta maioritariamente em linguagens de programação que servem de base para a arquitectura de *software* e de outras linguagens (meta-linguagens) sempre no intuito de facilitar a criação de soluções ou resolução de problemas ao utilizador/cliente. Com linguagens existentes criamos *frameworks* (Laravel, CakePHP) que oferecem um aglomerar de recursos que a linguagem sozinha não proporciona, além de uma sintaxe ou denominados "ajudantes" (*Helpers* [22]) que facilitam a programação de *software*. A um nível de *software* que facilita a criação de *software* (meta-software) temos à nossa mão editores de código (Brackets [23], SublimeText [24]) e IDEs (PhpStorm [25]) com *autocomplete*, funcionalidades para a depuração de erros, bibliotecas e pacotes que nos facilitam a vida. Todavia, há uma oferta demasiado grande e com a celeridade que a programação avança, a obsolescência está constantemente presente (Lawlor, 2015).

[21] *Brick and mortar*: expressão usada para descrever uma loja ou negócio tradicional comparativamente a um negócio somente presente na Web. http://www.learnersdictionary.com/definition/brick-and-mortar

[22] Helpers: auxiliam o programador na criação de tarefas. Cada *helper* traduz-se num aglomerado de funções procedimentais de uma categoria em particular — URLs, Cookies, Texto, entre outros. Fazem parte, usualmente, de *frameworks*.

[23] Brackets: http://brackets.io/.

[24] SublimeText: http://www.sublimetext.com/.

[25] JetBrains PhpStorm: https://www.jetbrains.com/phpstorm/.

2.1.7 Conclusão

Desde os primórdios da Computação e principalmente desde o nascimento das primeiras linguagens de programação de alto nível, houve um interesse pela parte de investigadores, estudantes e professores das Humanidades em automatizar resoluções de problemas que outrora encaravam manualmente. O laço entre as Humanidades e Computação é estabelecido através do trabalho vanguardista de Roberto Busa, inventor do hipertexto, que crava um novo paradigma nas paredes das Humanidades. Consoante a evolução da Computação, e paradoxalmente ao desenvolvimento de linguagens de alto nível para não programadores, o conhecimento computacional vai ganhando raízes no seio das Humanidades.

Ao mesmo tempo, os estudos em Computação começaram a ser integrados nos cursos de Humanidades, e surgiram conferências e publicações que permitiram difundir os mais recentes conceitos e debater ideias entre investigadores. Nos finais do séc. XX, assistimos ao nascimento da Internet que vem romper os canônes comunicacionais e introduzir o paradigma dos novos meios de comunicação que Manovich introduz, na viragem do século, com o seu livro visionário. Após *The Language of New Media*, a Web rapidamente se apropriou do dia-a-dia do ser humano, despoletando a existência de uma aldeia global em que a comunicação instantânea é possível em qualquer ponto do mundo. Actualmente, há todos os meios para um investigador e estudante pertencer ao seio da Computação e fruir da imensidão de ferramentas de concepção de obras de arte que o digital proporciona.

2.2 As Linguagens de Programação na Literatura Electrónica

Nesta secção são abordadas e analisadas as linguagens de programação que estiveram directamente ligadas à concepção de artefactos de literatura electrónica, desde os seus primórdios até à actualidade, pelos engenheiros informáticos que trabalharam com Pedro Barbosa. O estudo de caso aqui abordado pretende refletir a relação entre as Humanidades e a Computação. Serve de especial relevância perceber de que forma os investigadores em Humanidades lidaram com o advento da Computação e como é que fizeram uso das potencialidades da programação para produzirem obras de arte literárias que assumem contornos antes impossíveis na literatura tradicional.

2.2.1 Introdução

A evolução das tecnologias digitais pode ser fragmentada em quatro fases: 1) paradigma digital, 2) convergência, 3) electrónica de matéria sólida e, por último, 4) interface computador-máquina. Na primeira fase, dá-se o início da consciencialização da existência de um sistema binário, da codificação da informação e da Computação; mas para além da aritmética binária, o uso da lógica binária para operar a máquina, codificar instruções para dispositivos e códigos binários para transmitir informação. De acordo com (Ceruzzi, 2012), "este entendimento remonta a George Boole em 1854, ou precedentemente, a Gottfried Wilhelm Leibinz (1646-1716), que eram apologistas da teoria que os métodos digitais prevaleciam sobre os analógicos".

A segunda fase compreende que o termo Computação é representante de uma sinergia que contempla uma multitude de técnicas, dispositivos, e máquinas (por exemplo: rádio, televisão, computador, telefone, *smartphone*). Por outro lado, o computador espelha a convergência de outras tecnologias: dispositivos que calculam, armazenam informação e comportam a automação de tarefas. Resultando na criação de aparelhos digitais que foram gradualmente retirando o lugar ao analógicos.

A *electrónica de matéria sólida* teceu os seus inícios no séc. XX, aquando da evolução gradual tecnológica da electrónica; nos anos 60, com o advento da electrónica de matéria sólida, despoletou-se uma evolução fulminante. Gordon Moore, na sua lei, "expressava uma observação empírica datada em 1965 que a capacidade de armazenamento dos chips de memória de um computador estava, num ritmo estável, a duplicar a cada 18 meses" (Ceruzzi, 2012). Não obstante a apreciação de Moore estar intrinsecamente associada ao computador, surtiam ramificações que sofriam do mesmo sintoma, nomeadamente: capacidade do disco rígido, velocidade de processamento nos computadores e nas linhas de comunicações ou *bandwidth*. Com esta linha de raciocínio podemos concluir que os avanços tecnológicos historicamente resultam numa espada de dois gumes: por um lado, a evolução tecnológica molda a história; por outro, tem consequências políticas e sociais que culminam em alterações no epicentro da sociedade.

Chegando ao término destas etapas, deparamo-nos com a *interacção pessoa-computador* ou *user interface*. Tal como o termo explicitamente indica, há a troca de impressões entre o humano e a máquina. Contudo, tal como no parágrafo precedente, emerge um dilema: estaremos a criar sistemas que substituam o ser humano *ou* ferramentas que operem em harmonia com o seu criador e sejam simples extensões das suas faculdades mentais? Caminhamos a passos largos para a última, à medida que os programadores criam sistemas que conseguem responder ao humano em tempo real e na sua linguagem mãe (a título de exemplo: Microsoft Cortana [26]); O conceito *interacção pessoa-computador* abarca uma área vasta de conhecimentos que vão desde "as implicações filosóficas relacionadas com a humanidade a questões da arquitectura da máquina" (Ceruzzi, 2012).

O envolvimento por parte de estudantes e investigadores das Humanidades no advento da Computação, esteve primariamente vigente nas primeiras linguagens de alto nível que aqui são abordadas: Fortran; BASIC; Algol60; C++; Java; JavaScript. Nesta panóplia de linguagens não há preferências em quais seriam as mais indicadas para o desenvolvimento de um dado projecto porque essa decisão está relacionada com diversos factores: o orçamento; a experiência do programador; que linguagens estão disponíveis; compatibilidades entre máquinas; limite de tempo disponibilizado; entre outros. Com isto, "qualquer investigador de Humanidades, que tem acesso a linguagens de programação de alto nível não deve der dissuadido pela opinião de outrem que certa linguagem não é a mais indicada para um dado projecto. Todas as linguagens podem ser adaptadas para cumprir com determinados objectivos" (Raskin, 1971).

[26] Um assistente inteligente pessoal desenvolvido pela Microsoft desde 2009 e implementado em 2014 nos modelos de *smartphones* Windows Phone 8.1, nas braceletes Microsoft Band e futuramente no SO Windows 10. Cortana comporta as capacidades de criar lembretes, reconhecer voz humana e responder a questões orais usando o Bing — http://www.microsoft.com/en-us/mobile/campaign-cortana/

2.2.2 Fortran

Fortran foi a primeira linguagem de programação pensada para romper e revolucionar o paradigma da programação e as práticas na concepção de programas sem o auxílio de linguagens baseadas em *assembly*. A visão de John Backus partia da ideia de simplificar o código ao criar uma sintaxe mais simples e um número de comandos que permitiam a resolução de problemas com menos linhas de código; além disso, foi conceptualizada para ser suficientemente atraente e acessível para não programadores aprenderem a escrever código e a criarem soluções para problemas.

Introdução

Os inícios da linguagem de programação de alto nível Fortran (Formula Translating System) remontam ao verão de 1954 quando John Backus e sua equipa na IBM decidiram iniciar este projecto vanguardista. Findados os anos de desenvolvimento, Fortran foi comercializada em 1957. O objectivo da concepção de Fortran reside na "redução por um grande factor a tarefa de preparar problemas científicos para o 704, o próximo computador de grande porte da IBM" (Backus et al., 1957). Havia a necessidade de o 704 codificar problemas e programas automaticamente, com a ausência de erros e sem a presença humana. Era sabido que dois terços do custo para resolver a maioria dos problemas científicos e de engenharia em computadores grandes consistia no problema da preparação. Mais do que 90% do tempo para um problema era despendido no planeamento, escrita e depuração de erros nos programas. Nisto, Fortran "ajudou a abrir a porta para a Computação moderna" (Fortran, 2015), tornando o código de programação acessível a pessoas de outras áreas.

O Início da Revolução nas Práticas de Programação

O objectivo primordial do projecto Fortran era permitir ao programador especificar um procedimento numérico através do uso de uma sintaxe própria do sistema para obter, desta especificação, um programa eficiente no 704 que pudesse correr o procedimento. Era expectável que o resultado deste sistema reduzisse o custo de tempo num quinto na codificação e depuração de erros.

Após dois anos e meio de desenvolvimento, o sistema fora concluído, e comercializado, e desdobrava-se em duas componentes: a linguagem de programação Fortran e o tradutor para o 704 que consistia em converter programas codificados em Fortran para programas que corriam nos parâmetros do IBM 704. Os membros constituintes do projecto viram os seus objectivos alcançados ao constatarem que tinham reduzido o factor da tarefa da preparação do problema e aumentado a eficiência das saídas nos programas. Consequentemente, Fortran reduziu de 1000 instruções máquina-computador para 47 instruções automatizadas. A principal missão da linguagem estava adjacente ao aumento da eficiência de um programa computacional, colocando a questão da sintaxe como objectivo secundário. Porém, ambos, tiveram um papel primordial e revolucionário. Com isto, Fortran "introduziu comandos como o GO, TO e DO — propostos por

Harlan Herrick" (Fortran, 2015), que facilitavam a escrita dos programas através do uso de inglês simples.

Para um programador aprender Fortran, segundo os seus arquitectos (Kurtz, 1978), era necessário assistir a um curso de 1 dia e usar algum tempo para consultar o manual da linguagem. Consequentemente, o indivíduo, para executar um dado trabalho em que usou, durante 4 horas, 47 instruções do Fortran que foram posteriormente compiladas por um 704 em 6 minutos — produzindo 1000 instruções. Contudo, o programador constatou que o *output* saiu com erro, após ter corrido o programa. Nisto, o indivíduo observou o *output* e localizou o erro numa das instruções Fortran que tinha utilizado. Sumariamente, estimaram que este montante de trabalho demoraria 3 dias para codificar de fosse à mão, mais um tempo indeterminado para depurar erros. Thomas Eugene Kurtz, professor de matemática e cientista da Computação no Dartmouth College, conta que quando Fortran emergiu, pairava uma grande reticência no seio académico e refere que num projecto em que usava SAP (Symbolic Assembly Program), após meses de depuração e um total de 1 hora de funcionamento do IBM 704 para calcular a percentagem de pontos estatísticos num vector de valores de terceiro e quarto momento, declarou desistência. Kurtz decidiu tentar novamente mas desta vez em Fortran e em apenas 5 minutos do 704, solucionou o problema: "Esta lição, que programar em linguagens de alto nível pode poupar tempo computacional e humano, impressionou-me bastante" (Kurtz, 1978).

Influências de Outras Linguagens

Nos finais de 1953, John Backus propôs à IBM a concepção de uma linguagem de fosse menos complexa e menos morosa de ser usada, comparativamente com as em *assembly*.

Contrariamente à maioria das linguagens de programação, Fortran não foi alvo de qualquer influência em particular visto que se trata de uma linguagem revolucionária e a primeira de alto nível. No entanto, a linguagem "deriva de formulas algébricas usadas por cientistas e engenheiros que estavam presentes nos mais recentes avanços da tecnologia computacional" (Fortran, 2015).

Fortran nas Humanidades

Um dos exemplos mais pragmáticos da presença das Humanidades na Computação é do COCOA, um programa desenvolvido em Fortran, nos finais de 60 e inícios de 70, pela *Atlas Computer Laboratory* para o processamento de textos de linguagens humanas. Esta combinação de uns e zeros permite a produção de contagens de palavras e vocabulário, concordâncias com formatos de contexto e referências de localização, e traça perfis de frequência de palavras. De acordo com (Corcoran, 1974), o programa COCOA "suporta alfabetos até 256 letras de um caractere por letra, 128 letras de dois caracteres por letra, e 85 letras de três caracteres por letras". Pelo programa são aceites todas as línguas com excepção de caracteres e alfabetos não-latinos que terão que ser transliterados através de caracteres e combinações de caracteres compatíveis e presentes no sistema do computador.

Num segundo exemplo, o TEXTPACK é um processador de texto desenvolvido nos anos 70 por Juergen Hoeche, Hans-Dieter Klienemann e Klaus Radermacher que possui uma panóplia de funcionalidades similares ao COCOA. Mais tarde, "Peter Mohler e Conelia Zuell produziram o TEXTPACK V que possuía melhoramentos significativos e corria em PCs" (Olsen, 1989).

Por último, os trabalhos de literatura gerada por computador de Pedro Barbosa, em parceria com Azevedo Machado, intitulados por Permuta e Texal foram desenvolvidos em Fortran no LACA da FCUP (ver Secção2.4.2).

Conclusão

Antes da revolução computacional instaurada por Fortran, toda a programação era feita em linguagem-máquina. A Fortran, como linguagem de programação de alto nível, procurou substituir linguagens de baixo nível para diminuir significativamente o tempo nos processos de pensar, *bookkeeping* e escrever problemas num rácio estimado de 4 em 20. A possibilidade de ensinar Fortran a não programadores, como por exemplo historiadores ou investigadores em Humanidades, despoletou um grande interesse fora das ciências computacionais. No processo de depuração de erros, Fortran transmite uma quantidade substancial de informação para o utilizador determinar mais rapidamente, comparativamente com *assembly*, a fonte dos erros que constam no *output*. Fortran, em si, fornece ao programador informação suficiente para evitar código mal escrito. Desde a sua primeira versão em 1957, a linguagem "tem dominado várias áreas científicas durante muitas décadas" (Pao, 1999) e evoluído continuamente para as versões: II, III, IV, 66, 77, 90, 2003 e 2008. Passados quase 60 anos desde a sua implementação, "a primeira linguagem computacional de alto nível permanece em uso: no radar Doppler para previsões de tempo; em estudos atmosféricos e oceânicos, entre outros" (Fortran, 2015).

2.2.3 Algol

Perante o paradigma de linguagens de programação de alto-nível iniciado por John Backus com a invenção de Fortran, Algol vem reforçar a revolução nas práticas de escrever programação e vem proporcionar, com o desenho de Algol58, um modelo rico a seguir pelas linguagens sucedâneas — e, mais tarde, com Algol60, a afirmar-se no seio da Computação.

Introdução

O termo Algol advém de ALGOrithmic Language. Tanto Algol como Fortran são linguagens algébricas e partilham do mesmo *modus operandi* — foram arquitectadas para a resolução de problemas numéricos. Comportam variáveis, cadeias de vectores, declarações, instruções, instruções condicionais e iterativas, e procedimentos. Porém, Algol tem a vantagem de ter sido desenvolvida após Fortran, daí exibindo avanços significativos na arquitectura da linguagem comparativamente com Fortran; demonstrava melhoramentos na simplicidade de código, na sua estrutura e concisão (Wexelblat, 1981).

Um Modelo a Seguir

O desenvolvimento de Algol entre 1956 e 1957 despoletou uma revolução no *modus operandi* de programar. A primeira versão, Algol58, fora disseminada em 1958 e, paralelamente a Fortran, iniciou um novo paradigma nas ciências da Computação — rompendo com as normas de escrever código que na altura eram adoptadas nas linguagens em *assembly*. Por consequência, mais linguagens foram arquitectadas; não obstante de terem alcançado menos sucesso, foram igualmente relevantes para reforçar o novo movimento de produção de programas com a máquina, o computador — a título de exemplo: JOVIAL[27], MAD[28], NELIAC[29] e Euler[30]; e extensões: FORMULA ALGOL[31] e LCC[32].

A Algol58 seguia um padrão de princípios que era visto por muitos como ideal para a criação de linguagens de programação. Como explica (Perlis, 1978), Algol58 "sugeria possibilidades, mas ao mesmo tempo ignorava problemas dos quais qualquer implementação não pode contornar". Deste modo, a intenção de Algol58 residia na sua afirmação como um esqueleto que fosse usado como modelo para a criação de novas linguagens ou que fosse alvo de melhoramentos para eventualmente emergirem novas versões de Algol como mais tarde se evidenciou.

Um ano depois, em 1959, após uma análise profunda a Algol58, foram detectadas e corrigidas lacunas — e implementados melhoramentos. A estrutura de Algol58 permaneceu imaculada e Algol60 nasceu — desta vez, afirmando-se como uma linguagem completa e devidamente implementada. O seu impacto foi imediato, atraindo todos os tipos de especialistas em Computação: criadores de tradutores, analistas numéricos, *designers* em Computação, professores e estudiosos em linguagens (Perlis, 1978).

[27] JOVIAL ou Jules Own Version of the International Algorithmic Language fora desenvolvida por Jules Schwartz, em conjunto com a sua equipa, nos finais de 1958. As principais influências para a criação da linguagem foram a International Algebraic Language e SAGE. Os seus objectivos residiam na criação de uma versão algébrica de uma linguagem de programação e no desenvolvimento de dois compiladores para a Strategic Air Command em Omaha para que fossem usadas para múltiplas funções (Schwartz, 1978).

[28] MAD ou Michigan Algorithm Decoder é um programa de computador que traduz instruções algorítmicas para instruções equivalentes em linguagens-máquina. Algol58 foi usada como padrão para o desenvolvimento de MAD. A primeira tradução foi executada, com sucesso, em 1959 para o computador IBM 704 (Arden et al., 1966).

[29] NELIAC é um compilador baseado em Algol e fora desenvolvido pela U.S.Navy Electronics Laboratory como um compilador experimental, de inicialização (*"bootstrap"*), para o computador Remington Rand Univac Countess (Masterson, 1960).

[30] Euler foi uma linguagem de programação criada por Niklaus Wirth e Helmut Weber na Universidade de Stanford pensada como uma extensão de Algol60 — considerando melhoramentos ao nível da sintaxe (mais simples), em termos de eficiência e flexibilidade. Em suma, traduziu-se na criação de uma relação rigorosa entre a estrutura e o significado. A estrutura da linguagem sendo definida pela estrutura frásica da sintaxe; o significado, dizendo respeito aos efeitos em que execução da sequência da interpretação de regras exerce sob um conjunto de variáveis fixas denominado por ambiente (Wirth and Weber, 1966).

[31] FORMULA ALGOL é uma linguagem de manipulação de fórmulas, desenhada para Algol, para facilitar o processamento da linguagem. Neste sentido, FORMULA ALGOL é uma extensão para Algol que possuí funcionalidades tais como a manipulação de fórmulas, processamento de listas e a capacidade de processamento limitada a cadeias de caracteres (Perlis et al., 1965).

[32] LCC ou Language for Conversational Computing é uma linguagem de tempo compartilhado que permite aumentar a eficiência num sistema computacional e, ao mesmo tempo, permitir uma comunicação eficiente entre o programador e seus programas — comunicação que podemos denominar por conversação. LCC obteve maior sucesso em linguagens como JOSS e APL em detrimento de Algol, Cobol, Fortran ou PL/I (Mitchell et al., 1967).

Numa fase posterior, Niklaus, inventor da linguagem Pascal, era apologista da ideia que Algol60 necessitava de complementos para viabilizar outros objectivos além da Computação numérica. Com isto, a *International Federation for Information Processing* (IFPI) adquiriu um conjunto de pessoas e criou um *Working Group* com o intuito de desenvolver uma nova versão de Algol 60. Wirth fora escolhido para integrar o *Working Group 2.1* em 1964, no seio do qual surtiram contínuas discussões que enfatizavam a arquitectura da linguagem, definição de métodos formais, sintaxes, sequências de caracteres, entre outros, que resultaram numa ausência de consenso no seio da equipa. Após decorrerem outras tantas reuniões, o grupo fragmentou-se em dois: os ambiciosos que não tinham medo de arriscar e construir funcionalidades das quais desconheciam as consequências das suas implementações; os pragmatistas que entendiam que o significado de qualquer conceito era a soma das suas consequências, logo, estavam dispostos a criarem funcionalidades que permitissem a extensão de Algol 60 — tendo em mente o sucesso das implementações. Em 1965, o Working Group pediu a Wirth a submissão de uma proposta que visasse expor as opiniões dos pragmatistas. Contudo, numa reunião em Dezembro, a proposta do antigo membro de Algol 60, Adriaan van Wijngaarden, foi a seleccionada pela maioria para a Algol X — extensão da Algol 60. Independentemente da adversidade dos acontecimentos, Wirth implementou pelas suas mãos, a sua rejeitada proposta, chamando-lhe Algol W. A extensão de Algol 60, por Wirth, "passava a possuir novos tipos de dados que representavam pontos flutuantes e números complexos de dupla precisão" (Wirth, 1993). Apesar dos esforços, a implementação acabou por ser demasiado complexa e falhou como uma linguagem adequada para a criação de programas. Corridos uns anos, em 1968, Niklaus torna-se professor em Zurique no Federal Institute of Technology (ETH) onde "Algol 60 tinha sido a linguagem escolhida pelos investigadores em Computação numérica" (Wirth, 1993).

Influências de Outras Linguagens

A principal influência na arquitectura de Algol adveio de Fortran I (1956), a primeira linguagem de alto nível e de segunda geração. O destino de Fortran era visto por muitos como uma linguagem para ser adoptada universalmente mas a sua ligação intrínseca à IBM e seu *hardware* tornou esse destino pouco plausível e houve a necessidade de criar mais linguagens que permitissem satisfazer os programadores a um nível global. De acordo com (Wexelblat, 1981), "Algol foi mais graciosa que os seus predecessores, como por exemplo, Fortran, Math-Matic [33], e IT".

Algol nas Humanidades

Comparativamente a Fortran, Algol possui características que lhe proporcionaram destaque no seio das Humanidades, nomeadamente a recursão (estruturada em blocos) e a possibilidade de inserção de expressões em diversas instâncias do código, enquanto que Fortran só aceitava simples variáveis ou somente constantes. Ambas as linguagens foram desenvolvidas para tratarem de

[33] Math-Matic foi desenvolvida por volta da mesma altura de Fortran e possuía a mesma capacidade funcional; porém o compilador de Math-Matic tinha uma estrutura de paginação que não era replicável — problema que fora endereçado mais tarde (Sammet, 1992).

variáveis alfabéticas que problemas de língua usualmente requerem. Algol distancia-se de Fortran por ser mais fácil para tarefas de documentação ao permitir instruções mnemónicas, blocos de etiquetas (*block labels*), entre outros (Raskin, 1971).

Pedro Barbosa entrou em contacto com Algol60 aquando das suas experiências com literatura electrónica no LACA, juntamente com Azevedo Machado, que deram azo à criação do Re-Text, Texal e Permuta (ver Secção 2.4.2).

Conclusão

O desenvolvimento de Algol foi particularmente importante para dar seguimento ao movimento iniciado por Fortran na arquitectura de linguagens de alto nível, que não estivessem somente acessíveis a programadores mas a interessados no desenvolvimento de soluções que fossem aplicáveis às suas áreas de estudo/investigação. Algol acaba por ser uma continuação do que Fortran iniciou, contemplando mais características num ambiente mais versátil. Destarte, Algol60 atingiu uma aceitação global no seio dos programadores e na disseminação da ideia que a programação deve ser usada por todos. Mais tarde, BASIC vem reforçar essa mesma ideia.

2.2.4 BASIC

Esta secção descreve os fundamentos da linguagem BASIC, uma das primeiras linguagens de alto nível, concebida por Thomas Kurtz. BASIC pretendia, como objectivo primordial, facilitar a aprendizagem de código a não programadores mas apesar de ter evoluído continuamente, não obteve grande sucesso. Serve de especial importância entender de que forma BASIC esteve presente nas Humanidades para a concepção de literatura electrónica.

Introdução

> "O objectivo primordial do BASIC é tornar a Computação acessível a mais pessoas" (Kurtz, 1978).

A linguagem BASIC (Beginners All-purpose Symbolic Instruction Code) nasceu nos meandros da Dartmouth College pelas mãos de John Kemedy e Thomas Kurtz em 1963/4. As suas primeiras edições foram substancialmente influenciadas pelas linguagens de Computação que lhe antecediam (exemplo: Fortran), BASIC assumiu com o seu crescimento, uma identidade própria. Os autores de BASIC tiveram como ideia principal que a linguagem fosse também usada por não programadores e "fosse manuseada interactivamente, sendo concebida para a criação e depuração de programas" (Lientz, 1976). As expectativas iniciais referentes à proliferação da linguagem, foram rapidamente ultrapassadas pelo sucesso da implementação de BASIC em escolas secundárias — expandindo posteriormente para universidades e indústrias.

Programação Acessível a Todos

A versão original de BASIC consistia em 14 comandos e partilhava, como outras linguagens vieram posteriormente a adoptar, do *modus operandi* de tempo compartilhado[34]. Como explica Thomas Kurtz em (Wexelblat, 1981), "[o]s 8 ou 10 comandos encarados pelos utilizadores como mais simples de manusear ficaram desde então associados ao BASIC". Assim sendo, a linguagem geria três tipos de informação: variáveis numéricas, variáveis de cadeias de caracteres, e cadeiras de vectores de uma ou duas dimensões de tipos de cadeia de vectores ou numérico.

Para o início da criação de um programa em BASIC era necessário chamar o comando NEW, que, por sua vez, tal como os outros comandos, estava documentado em LIST. Deste modo, o DELETE servia para editar o código, o SAVE para o guardar e, por fim, na descrição dos comandos base, o RUN para correr o programa. Kurtz reforça que "o BASIC é único no facto de o nome das variáveis estava limitado a uma letra ou dígito seguido do símbolo $ se fosse uma cadeia de caracteres" (Wexelblat, 1981).

Num estudo publicado no livro *Datapro* de 1975, que visava perceber em 101 organizações que linguagens de programação eram as mais usadas, concluiu-se que "Fortran era utilizada por 65% dos indivíduos — ficando o BASIC em segundo lugar, com 49%" (Corporation, 1975). A implementação do BASIC constava em 58 das 98 utilidades de tempo compartilhado que estavam disponíveis tanto em mini-computadores como em computadores de grande porte e que funcionavam dicotomicamente: em modo interactivo e em processamento em *batch*.

A premissa de que BASIC pode ser ensinado a não programadores é derrubada no estudo *A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements* por Bayman e Mayer. Os investigadores procuraram perceber em 1983 o que é que iniciantes em programação conseguem compreender e apreender após terem lições de BASIC. Segundo (Bayman and Mayer, 1983), "este estudo explora a ideia de que aprender BASIC involve mais do que a aquisição de factos específicos, regras, e competências". Há a necessidade de os aprendizes desenvolveram modelos mentais para a linguagem com vista a apreenderem as ramificações de BASIC. Perante esta problemática, os programadores podem ter desenvolvido um modelo de concepção correcto no uso das instruções, enquanto um iniciante poderá ter apreendido um modelo de concepção errado. No processo de investigação levado a cabo por Bayman e Mayer, seleccionaram uma amostra de 30 sujeitos sem um curso superior, sem experiência com computadores e sem conhecimento de programação computacional. Os 30 estudantes foram submetidos a um curso de 6 horas, dividido em 3 sessões, de BASIC em que tinham um Apple II, um manual de programação e uma versão modificada de BASIC — para instruir indivíduos, como recursos. Após a aprendizagem de 9 instruções, os estudantes tiveram um exame final como reconhecimento de aquisição de conhecimentos que contava com os seguintes 4 critérios: correcto (caso os sujeitos nos seus parâmetros incluíssem transacções chave correctas e nenhuma incorrecta — exemplo: "*LET A* =

[34]Tempo compartilhado ou *time-sharing* é a partilha de recursos computacionais por vários utilizadores simultaneamente por meios de multi-programação e multi-tarefa. A BASIC foi a primeira linguagem a contemplar este método que permitia o uso de um computador a pessoas sem que possuíssem um, através do acesso a um só computador. Esta fragmentação de recursos permitiu cortar significativamente em custos e fomentou a criação de aplicações interactivas.

B +1 seria desdobrado em "guarda o valor de B mais 1 no espaço de memória A" (Bayman and Mayer, 1983)); incompleto (se os sujeitos escrevessem transacções chave incompletas e nenhuma correcta ex: "*LET A = B +1* com a descrição guarda o valor de B + 1 na memória A" (Bayman and Mayer, 1983)); incorrecto (caso os indivíduos produzissem uma ou duas transacções incorrectas, por exemplo: "*LET A = B +1* descrito como guarda a equação *A = B + 1* na memória" (Bayman and Mayer, 1983); vazio (caso se verificasse que o estudante não produziu transacções correctas ou incompletas).

Esta investigação levou a concluir que aprender e praticar BASIC, individualmente, não é suficiente para iniciantes em programação porque há a tendência de interpretarem erradamente as instruções e absorverem informação errónea. Consequentemente, é essencial que o aprendiz tenha acesso a aulas com um professor para compreender as particularidades inerentes ao computador e como é que funciona internamente para posteriormente entender o uso de cada comando de BASIC na concepção de programas.

Apesar dos contributos que BASIC proporcionou ao universo das linguagens de programação e à concepção de programas, Edsger Dijkstra, cientista computacional de nacionalidade holandesa, critica severamente BASIC no seu livro *Selected Writings on Computing: A Personal Perspective* de 1982: "É praticamente impossível ensinar boa programação a estudantes que tiveram exposição prévia ao BASIC: como potenciais programadores, eles estão mentalmente mutilados para além da esperança de regeneração" (Dijkstra, 1982).

Influências de Outras Linguagens

A linguagem BASIC foi fortemente influenciada por Fortran II e ALGOL 60. Contava com algumas adições que permitiam uma maior eficiência na actividade de tempo compartilhado (*time sharing*). À medida que foi evoluindo, distanciou-se das suas origens e desenvolveu uma identidade própria.

BASIC nas Humanidades

Um dos exemplos de trabalhos em BASIC foi realizado por Pedro Barbosa e Azevedo Machado em 1988 (ver Secção 2.4.2), intitulado *Máquinas pensantes. Aforismos gerados por computador* — um gerador textual de poesia experimental (Barbosa, 1988).

A obra digital *First Screening* de 1983-84, desenvolvida por bpNichol em Apple II BASIC, dialecto de BASIC, é demonstrativo de um conhecimento de código em execução não só para ferramentas de análise de textos longos, assim como para a compreensão dos mesmos textos. Este artefacto de poesia experimental criado para o Apple II e disseminado através uma disquete, permitiu bpNichols "expandir a sua poesia normal e concreta de estática para uma forma em movimento através da sua exposição em ecrã" (Brown and Trowbridge, 2012).

Outras experiências com BASIC no âmbito da literatura electrónica e poesia experimental foram conduzidas por Silvestre Pestana na sua obra *Computer Poetry* entre 1981 e 1983. A par de

bpNichol, o contexto em que Pestana desenvolveu a sua obra insere-se na fase inicial dos computadores portáteis em que a obrigatoriedade de ter que aceder a edifícios institucionais onde residiam os computadores de grande porte deixava de existir. Deste modo, Silvestre Pestana programou nos computadores Sinclair ZX-81 e Sinclair ZX Spectrum três poemas, dedicados a Henri Chopin, E. M. de Melo e Castro e Julian Beck (Seiça, 2015).

Conclusão

A proliferação do BASIC e seus comandos foi veloz nos anos 60 e 70 em que a linguagem atingiu o seu auge. Em meados dos anos 70, "quase todos os sistemas pequenos ofereciam o BASIC, quer exclusivamente ou em junção com outras linguagens" (Kurtz, 1978). Pelos anos 80 quase todos os computados pessoais tinham um compilador de BASIC. Das suas origens, a linguagem tomou novos rumos e variantes do nome ao dar vida a QuickBASIC, uma linguagem que teve um curto tempo de vida, e ao Visual Basic, que era possuidora de uma interface gráfica que facilitava a concepção de programas.

2.2.5 C++

A linguagem C++, apesar de ter sido desenvolvida nos anos 80, moldou-se às evoluções tecnológicas e continua a ser abundantemente usada e relevante nos dias correntes. C++ tem um papel preponderante em todas as áreas em que sistemas binários operam e nas Humanidades teve e continua a ter um papel preponderante.

Introdução

A linguagem de programação C++ foi desenvolvida por Bjarne Stroustrup e disseminada em 1983 para proporcionar alicerces à Simula[35] para a organização de programas paralelamente com a eficiência e flexibilidade do C para a arquitectura de programas. O objectivo inicial na criação de C++ não contemplava inovação mas sim um aumento significativo nas componentes de eficiência e flexibilidade. Não obstante, com o decorrer da evolução da linguagem, surgiram implementações que se traduziram em inovação, reformulação e melhoramentos a nível de sintaxe — permitindo a C++ distanciar-se de outras linguagens; não comprometendo, contudo, a sua eficiência e flexibilidade (Bergin, 1996).

O Sucesso de uma Linguagem

No espaço de tempo entre 1982 e 1984, os objectivos da linguagem tornaram-se gradativamente mais ambiciosos, estratégicos e específicos. A C++ assumiu uma identidade própria consoante o interesse pela linguagem foi florescendo no seio dos programadores na Bell Labs. Deste

[35] A Simula é o nome para duas linguagens de programação de simulação: Simula I e Simula 67. A linguagem foi desenvolvida nos anos 60 na Noruega no Centro Computacional Norueguês por Ole-Johan Dahl e Kristen Nygaard. Simula é considerada a primeira linguagem orientada a objectos e foi desenhada para executar simulações de programas (Dahl et al., 1968).

modo, criaram-se bibliotecas, ferramentas para a resolução de problemas e, mais enfaticamente o compilador de *front-end* para C++ intitulado Cfront[36]. Paralelamente à versão experimental *C with Classes*, o sucesso de C++ foi imediato. Com isto, assistimos a um afastamento das suas origens inerentes a C para uma linguagem que assume características e funcionalidades únicas.

Corria o ano de 1986 a meio quando a versão 2.0 de C++ é lançada com três importantíssimas implementações: tipos de parametrização, herança múltipla[37] e tratamento de excepções[38]"O sistema de tratamento de excepções foi desenvolvido entre 1984 e 1989".

Na altura do lançamento da segunda versão, a linguagem contava com um número estimado de dois mil utilizadores a nível mundial. A agenda de trabalhos, após a introdução de C++ 2.0, em conjunto com Steve Johnson, implicava que o desenvolvimento e suporte diário de ferramentas estivesse incumbido a um grupo de programadores para que Johnson e Stroustrup se pudessem focar no desenvolvimento de novas funcionalidades e suas bibliotecas. Paralelamente, era mesmamente expectável que a AT& T, simultaneamente com outros parceiros, iniciassem a conceptualização e implementação de outros compiladores e ferramentas para a expansão de Cfront. No entanto, a despeito da definição do planeamento ter sido concluída, e de a AT& T já ter iniciado a resolução dos objectivos, o projecto foi comprometido pela incerteza e carência de concentração nos pontos centrais do planeamento pelos gestores do projecto. O lançamento da versão 1.3 do Cfront, planeada para os inícios de 1988, foi adiada — somente em Junho de 1989 é que assistimos a uma nova versão do Cfront, a 2.0. A maioria das pessoas que estiveram presentes no desenvolvimento de *C with Classes* e C++ continuaram a contribuir para a evolução da linguagem. A quantidade estimada de utilizadores de C++, a partir de 1986, disparou, "duplicando a cada sete meses e meio" (Bergin, 1996). Se em 1986 as estatísticas apontavam para 2 mil utilizadores, em 1991 atingiram os 400 mil (Bergin, 1996).

A primeira publicação inteiramente dedicada à linguagem, intitulada por *The C++ Report*, começou a ser disseminada em Janeiro de 1989, contando com Rob Murray como editor. Em paralelo, o *The C++ Journal*, com uma tiragem semestral, emergia em 1991.

Influências de Outras Linguagens

A C++ ascendeu de uma versão experimental intitulada *C with Classes*, desenvolvida entre 1979 e 1983 no Computing Science Research Center of Bell Laboratories em New Jersey, que consistia, como o nome sugere, da linguagem C mas com classes. Esta versão foi pensada para melhorar a organização de programas em C através da sua fragmentação em classes. O resultado da versão experimental e, consequentemente, do nascimento de C++, despoletou-se perante a

[36]Cfront é um compilador de *front-end* que executa uma análise do foro semântico e sintáxico da linguagem, criando, por sua vez, uma representação interna do *input* para ser reformulada e resultar num *output* que reúna todas as características para a geração de código (Bergin, 1996).

[37]Herança múltipla ou *multiple inheritance* é uma característica de certas linguagens de programação orientadas a objectos que consiste na possibilidade de um objecto ou classe se apropriar de informação ou características de mais do que um objecto pai ou classe pai

[38]As excepções tinham um papel relevante na depuração de erros em programas arquitectados com o auxílio de bibliotecas criadas fora da equipa de programadores de C++. De acordo com (Bergin, 1996)

tentativa de analisar o kernel[39] do sistema operativo UNIX "com fim a determinar até que ponto poderia ser distribuído numa rede local de computadores ligados entre si" (Bergin, 1996). Após o início de *C with Classes* em 1979, foi só durante 1982 que Stroustrup percebeu que a linguagem ou versão experimental se traduzia num sucesso parcial e que a sua expansão era uma impossibilidade uma vez que a sua atractividade no seio dos programadores e investigadores era insuficiente para que houvesse investimento financeiro de modo a permitir o seu desenvolvimento.

C++ nas Humanidades

A primeira versão do programa Sintext de Pedro Barbosa, foi programado em C++, por Abílio Cavalheiro, sendo uma das obras no seio das Humanidades (ver Secção 2.4.2).

A aplicação de C++ à biblioteca online e-Science é de particular relevância porque trata-se de um arquivo digital com artigos científicos e um sistema eficiente para a análise textual e pesquisa foi o intuito do projecto *The Researching e-Science Analysis of Census Holdings* (Terras, 2009).

Conclusão

A linguagem C++, inventada nos anos 80, tem vindo a manter-se relevante desde então, adaptando-se às conjunturas tecnológicas e primando pela sua versatilidade. A característica mais forte de C++ é a sua capacidade de operar num ambiente tradicional (em termos sociais e computacionais), o seu tempo de execução e eficiência de espaço, a flexibilidade no conceito de classes, o seu baixo preço, e a sua natureza não-proprietária. Os pontos mais fracos de C++, comparativamente a linguagens que surgiam a seguir, foram herdados de C (por exemplo: o pré-processador) (Bergin, 1996).

2.2.6 Java

A Java, em conjunto com a Smalltalk, foram revolucionárias pelas suas capacidades de ubiquidade de código e pela sua orientação a objectos. A existência da máquina virtual JVM permite que os programas corram em qualquer sistema operativo.

Introdução

O desenvolvimento de Java por James Gosling, juntamente com a sua equipa na Sun Microsystems, iniciou a 1991, culminando na primeira versão do projecto em 1995. Java opera com a licença GNU que viabiliza a sua livre manipulação. A linguagem era sinónimo de ubiquidade e de ambição, expressados no *slogan*: "escreve uma vez, corre em qualquer lado" (Louden, 2011). Inicialmente, tinha sido planeada para ser implementada em sistemas presentes em eletrodomésticos mas imediatamente perceberam que se cumprissem com os objectivos, Java poderia ser implementado em qualquer máquina.

[39] Kernel é um programa de computador que gere o *input* e o *output* dos pedidos de *software* e os traduz em instruções para processamento de informação na unidade central de processamento do sistema, assim como para outros componentes electrónicos de um computador.

A Primeira Linguagem Ubíqua

A linguagem de alto nível e orientada a objectos, para cumprir com o seu *slogan*, tinha que compilar *bytecode* (tipo de código binário) e teria que existir, num sistema pretendido, uma máquina virtual *Java Virtual Machine* (JVM) ou um interpretador para executar o *bytecode* do programa em Java. Com a linguagem, advinham a distribuição e desenvolvimento de *software* JVM que "estiveram em sintonia com os progressos e desenvolvimentos na Computação nos últimos 20 anos em redes, Web, dispositivos móveis, entre outros" (Louden, 2011). Os aquitectos de Java perceberam que a linguagem devia ser totalmente orientada a objectos. Para atrair a generalidade de programadores, era relevante possuir uma sintaxe similar às linguagens da época.

No circuito do desenvolvimento Web, Java obteve um célebre reconhecimento por parte de Web *designers* que pretendiam inserir animação e interactividade nas suas páginas, frequentemente alicerçadas em bibliotecas como a Swing[40]. A facilidade de leitura de programas em qualquer computador, pessoal ou de grande porte, é vista como uma mais valia. Deste modo, como acontece com o desenho subjacente à Internet, Java aproveitou o crescente poder de computadores pessoais e postos de trabalho para fornecer capacidades de tradução de código. O impacto de Java foi imediato mas o interesse dos média foi desproporcional e infiel às características inerentes ao Java, resultando na disseminação de informação errónea para a opinião pública (Ceruzzi, 2003).

Influências de Outras Linguagens

Como linguagem de programação, Smalltalk teve uma grande influência sobre os fundamentos de Java. A Smalltalk rege-se pelo princípio de ubiquidade e é orientada a objectos. Teve o seu início na década de 70 no *Xerox Palo Alto Research Center Learning Research Group* composto por uma equipa que trabalhava em função de um conjunto de objectivos: permitir que pessoas de diferentes áreas conseguissem usufruir com eficiência das funcionalidades que a Computação oferecia; desenvolver um sistema de informação poderoso em que o utilizador pode armazenar, aceder, e manipular informação para que este pudesse expandir em tamanho em concordância com o reconhecimento por parte do utilizador de como operar o sistema de uma forma eficiente. O grupo mudou de nome em 1981 para *Software Concepts Group* (SCG), e a primeira versão de Smalltalk começou a ser comercializada em 1980 — intitulada Smalltalk-80. A linguagem comportava "um ambiente de programação gráfico e interactivo. Foi desenvolvida com o propósito de que todos os componentes do sistema fossem acessíveis ao utilizador para que possibilitassem a observação e manipulação" (Goldberg and Robson, 1983).

Java nas Humanidades

O gerador textual de poesia experimental Sintext-W de 1998, desenvolvido por Pedro Barbosa e José Manuel Torres, é um dos exemplos de programas desenvolvidos em Java no seio das Humanidades.

[40] Swing é um interface com ferramentas gráficas parte do espólio da bibliotecas do Oracle.

Conclusão

A linguagem Java foi revolucionária pela sua ubiquidade e marcou o início da proliferação de *software*, visto que todos os SO necessitam de JVM para lerem o programa — permitindo que a premissa inicial da linguagem fosse alcançada com sucesso. Momentaneamente a Java encontra-se na versão 8 e é ainda das linguagens mais utilizadas nos meandros da programação. Nas aplicações Android existe como base de programação; porém, assenta numa variante de Java que possui bibliotecas díspares do Java puro, direccionadas a sistemas Android.

2.2.7 JavaScript

JavaScript (JS) como linguagem de *script* é correntemente uma das mais utilizadas no desenvolvimento Web para a criação de páginas dinâmicas, assim como para tratar da relação entre cliente e servidor. Nesta última secção, a linguagem é analisada e interligada com obras de literatura electrónica que nasceram ou foram convertidas em JavaScript.

Introdução

A JavaScript, linguagem em *script* derivada da ECMAScript, foi arquitectada por Brendan Eich da Sun Microsystems e ECMA International e viu a sua implementação no ano de 1997. Como afirma (Rauschmayer, 2014): "Sempre que há a referência a versões de JavaScript, emerge o nome ECMAScript que se encontra na quinta versão — transitando momentaneamente para a sexta". O propósito inicial da linguagem no meio do desenvolvimento Web cingiu-se à execução de pequenas tarefas; correntemente o seu papel assume contornos de especial relevância na forma de como a Web está a ser moldada dia após dia. Com a invenção de NodeJS por Ryan Dahl, em 2009, a JavaScript vai para além de uma linguagem de *front-end* e presentemente compete ao lado de PHP na gestão de informação entre o cliente e o servidor denominado por *back-end*. Não obstante, há até ao momento algumas restrições em *browsers* que inviabilizam a acção de *scripts* por medidas de segurança inerentes à possível transmissão de vírus.

A Linguagem em Voga no Desenvolvimento Web

Presume-se frequentemente que JavaScript é Java, quando ambas só estão relacionadas ao nível de marketing e diferem significativamente como linguagens. Como explica (Benson, 1999):"Enquanto que a Java é baseada em classes e instâncias, JavaScript tem um prototipo/instância[41] baseada num sistema de objectos em que carece da separação de classes e objectos". O sistema de objectos protótipo/instância é especialmente indicado para linguagens dinâmicas que são usadas para a criação de aplicações interactivas.

Por meios do HTML5 que representa conteúdo e do CSS3 que a apresenta, a JavaScript fica encarregue das acções ou comportamentos nas páginas Web. Consequentemente, a JS fomenta a adição de interactividade ao permitir o desenvolvimento de páginas dinâmicas (Duckett, 2014).

[41] Instâncias de objectos podem ter um ou mais protótipos que dos quais podem incumbir mensagens para aceder a comportamentos mais gerais e partilhados (Benson, 1999)

A JavaScript apresenta problemas de segurança pelo facto de que o espólio de objectos contido numa aplicação Web partilha do mesmo ambiente sem quaisquer limitações. Consequentemente, despoletam-se problemas de segurança adjacentes à capacidade de modificar, dinamicamente, a aplicação Web e ter acesso total aos objectos que lá residem. Com o intuito de contornar estas debilidades inerentes à linguagem, há a necessidade que o programador esteja informado acerca de métodos que garantam a segurança da aplicação Web. Um desses métodos, a título de exemplo, está inerente à declaração de variáveis locais, em detrimento das globais, (var $myLocalVariable = arg + 4$;) que são só acessíveis localmente e pelo corpo da função que as envolve (Ducasse et al., 2012).

A popularidade de JavaScript tem vindo a aumentar exponencialmente nos últimos 5 anos, especialmente depois de Ryan Dahl ter desenhado e implementado uma estrutura para a JavaScript tratar de pedidos entre o cliente e o servidor (*back-end* — competindo directamente com PHP e Ruby), antes impossível, ao inventar NodeJS[42] em 2009. Posteriormente surgiu o MEAN stack[43], um conjunto de *software* alicerçado no NodeJS que contempla: AngularJS, *framework* em MVC que trata do *front-end*; Express, uma *framework* minimalista e flexível; e, por último, a base de dados NoSQL[44] MongoDB [45]. Dentro do espectro das *frameworks* já mencionadas, surgem a Meteor[46] e a EmberJS[47], ambas recentes e em contínua evolução. Uma das vantagens em usar *frameworks* consiste não só na panóplia de componentes que estão embebidas por omissão mas assim como características inerentes à segurança das aplicações e a um conjunto de princípios que as *frameworks* obrigam os utilizadores a seguirem para assegurarem a robustez dos seus produtos.

Momentaneamente, a ECMAScript, a linguagem mãe da JavaScript, está no processo de transição da versão 5, que remonta a 2009, para a 6[48], contemplando uma série de melhoramentos ao nível de classes, objectos literais, geradores, entre outros.

JavaScript nas Humanidades

Um dos exemplos pragmáticos do uso de JavaScript nas Humanidades é de Judd Morrissey em *The Last Performance* (ver Figura 2.5a) que serve como resposta ao trabalho seminal colectivo de performatividade dos Goat Island [49], que após 20 anos, fizeram a sua nona e última actuação intitulada de *The Lastmaker* em 2007 (Morrissey, 2007a). Deste modo, Morrissey com a sua peça, conceptualizada para o ambiente Web, convida os participantes a contribuir com texto para

[42] NodeJS: https://nodejs.org/

[43] MEAN stack: http://mean.io/

[44] NoSQL (Not only SQL) surgiu com as aplicações Web que se posicionam na Web 2.0 e possuem um grande quantidade informação em bases de dados, como são os casos das redes sociais e grandes empresas. Como solução para a perda de desempenho numa aplicação Web que um número crescente de utilizadores causa, emergiu a necessidade de escalar a informação horizontalmente; isto é, tentar aumentar o desempenho através da aumento de unidades de armazenamento (Abramova and Bernardino, 2013).

[45] MongoDB: https://www.mongodb.org/

[46] Meteor: https://www.meteor.com/

[47] EmberJS: http://emberjs.com/

[48] ECMAScript 6: https://github.com/lukehoban/es6features/blob/master/README.md

[49] O trabalho de performatividade do grupo agora inactivo "é uma série de respostas aos desafios a que nos entregamos, ao que acontece no nosso redor, aos acontecimentos do mundo, mas maioritariamente respostas entre uns e outros" (Christopher et al., 2007).

(a) *The Last Performance* por Judd Morrisey

(b) *First Screening* de bpNichols

Figura 2.5: Obras de Judd Morrissey e de bpNichol em JavaScript

os campos de entrada que vão surgindo no decorrer do programa. A sua obra de arte aglomera conceitos de "escrita colaborativa com restrições, arquivo, e visualização textual para responder ao tema de intemporalidade correspondente às formas aquitectónica, acções de construção, actuação final, e a interrupção de uma comunidade" (Morrissey, 2007b).

A última versão do programa Sintext de Pedro Barbosa, codificado originalmente em C++ por Abílio Cavalheiro e em Java por José Manuel Torres, foi re-programado para JavaScript por Nuno Ferreira com o *script* Poemario.js. Destarte, é possível manter o programa vivo e a operar de modo ubíquo, por um lado ao permitir uma preservação da obra digital pela sua actualização mediante a conjuntura tecnológica em que vivemos e, por outro lado, a consequente fuga à obsolescência tecnológica.

O artefacto de poesia experimental *First Screening* de bpNichol (ver Secção 2.2.4), originalmente escrito em Apple BASIC, é outro exemplo de uma obra re-escrita por Marko J. Niemi e Jim Andrews para uma linguagem globalmente aceite pelos Web *browsers* (ver Figura 2.5b).

Influências de Outras Linguagens

O criador de JavaScript, Brendan Eich, sentia a pressão para criar uma linguagem antes que a Netscape arquitectasse uma que fosse prontamente adoptada e deixasse JavaScript na sombra. Para isso, Eich partiu de uma panóplia de linguagens existentes para criar um puzzle à sua medida. De Java fora extraída a sua sintaxe, do Scheme[50] e AWK[51] as suas funções de primeira classe, do Self[52] a inerência prototipada, e por último: cadeias de caracteres, variáveis e expressões comuns

[50] Scheme foi a primeira linguagem a incorporar procedimentos de primeira classe como o cálculo de lambda, daí proporcionando a usabilidade de regras de *scope* estático e uma estrutura em bloco numa linguagem escrita dinamicamente. A Scheme influenciou em grande parte o desenvolvimento de Lisp (Clinger, 1987).

[51] AWK é uma linguagem de programação, e sistema, criada em 1977 por Alfred Aho, Peter Weinberg e Brian Kernighan. As suas características fazem com que seja uma ferramenta útil para a resolução de problemas. A AWK tem a capacidade de processar ficheiros e interligá-los aos registos e valores de *input*, além de estabelecer uma correspondência de padrões nesses ficheiros (Levy, 1983).

[52] Self é uma linguagem de programação orientada a objectos, desenvolvida por Randall Smith e David Ungar em 1987, baseada em objectos.

provenientes de Perl e Python. Com base no seu conjunto de suas influências, evidenciámos uma mistura entre programação funcional e programação orientada a objectos (Rauschmayer, 2014).

Conclusão

De acordo com (Wirth, 1993), "o principal papel de um criador de linguagens de programação passa por ser o de um curador e coleccionador activo de conceitos e características. Quando estes conceitos forem seleccionados, formas de os expressar devem ser encontradas — por exemplo: a sintaxe deve ser definida. As formas de expressar conceitos individuais devem ser cuidadosamente moldadas num todo. Isto é o mais importante, caso contrário a linguagem aparecerá incoerente." Enquanto que a maioria das linguagens orientadas a objectos ficam na gaveta, com excepção de Java e C++, a JavaScript é actualmente uma das linguagens mais populares e versáteis — assumindo posições em *front-end* com o AngularJS, EmberJS, entre outros, e com *back-end* como é o exemplo do NodeJS. Não obstante das características mencionadas, a JavaScript inseriu-se num nicho como linguagem *script* que trata da relação entre cliente-servidor, antes impossível (Rauschmayer, 2014).

2.2.8 Conclusão

O estudo das linguagens de programação de alto nível que estiveram no advento da Computação e permitiram a investigadores externos às ciências da Computação aprenderem programação foram centrais neste estudo. A possibilidade de artistas e investigadores aumentarem a riqueza dos seus trabalhos, independentemente da área de estudos, e diminuírem o tempo que demoravam a executar uma dada tarefa manualmente, vem revolucionar a investigação e dar azo à criação de artefactos de literatura electrónica em contornos antes impossíveis pelas ferramentas tradicionais.

2.3 A Literatura Electrónica

Esta secção procura definir o paradigma da literatura electrónica e explicar a sua evolução segundo investigadores que participaram activamente no desenvolvimento deste fenómeno que redefiniu os cânones da literatura.

2.3.0.1 Introdução

A origem do estudo dos média remonta aos meados dos anos 60 do séc. XX. O visionário Marshall McLuhan, frequentemente conotado como sendo o *Oráculo da Idade Electrónica*, introduziu o conceito de *aldeia global*, em que daí advinha a frase: "o meio é a mensagem" — profetizando o impacto das tecnologias electrónicas na civilização. Deste modo, instituiu o campo a que hoje atribuímos como sendo o estudos dos média.

O termo *literatura electrónica* (*e-lit*) é geralmente empregado para descrever uma parafernália de práticas literárias computacionais. A *Electronic Literature Organization* (ELO) descreve-a

como sendo relativa a "obras de especial relevo literário que tiram partido das capacidades e contextos providenciados pelo computador isolado ou ligado em rede a outros" (ELO, 2015). Scott Rettberg simplifica o seu significado, afirmando que a "literatura electrónica é o resultado ou produto da actividade literária realizada no computador" (Rettberg, 2014).

2.3.0.2 O Paradigma da Literatura Electrónica

> "A literatura electrónica não é somente uma 'coisa' ou um '*medium*' ou sequer uma estrutura de 'trabalhos' de vários 'géneros' — é, indiscutivelmente, um modelo cultural emergente, tanto quanto uma colecção em colectivo de termos, palavras-chave, géneros, estruturas e instituições na produção de novos objectos literários" (Tabbi, 2010).

No seguimento deste raciocínio, a literatura electrónica, em traços gerais, exclui as digitalizações — objectos de sua origem analógica, posteriormente convertidos para uns e zeros, e apela aos objectos nascidos no seio digital: criados e lidos pelo computador. Os leitores de literatura tradicional interagem com o digital comportando as expectativas formadas pelo impresso, "inclusivamente o extenso e tácito conhecimento de modelos de letras, convenções de impressão, e modos de impressão literária" (Hayles, 2008). Com a finalidade de a literatura electrónica se afirmar como um paradigma emergente e revolucionário, deve partir da existência das expectativas inerentes à literatura tradicional e transformá-las ou modificá-las para que encaixem nos moldes da *e-lit*.

A primeira geração de *e-lit* não contou prontamente com a implementação do hipertexto e da interactividade mas na curva de evolução do paradigma surgiu a primeira obra hipertextual da autoria de Michael Joyce e intitulada *Afternoon: a story* (1987). Paralelamente à primeira ficção hipertextual de literatura electrónica, Joyce introduziu o *software* Storyspace, um sistema hipertextual — "desenvolvido para funcionar como um ambiente de escrita para o desenvolvimento de obras hipertextuais por escritores experientes e profissionais" (Joyce, 1991).

Pedro Barbosa fundou em 1991 o Centro de Estudos sobre Texto Informático e Ciberliteratura (CETIC) na Universidade Fernando Pessoa, célula de investigação para a reflexão teórica e prática crítica "das novas modalidades de texto nascidas com o advento da informática e da novas tecnologias digitais — em particular: o texto automático, o texto dinâmico e o hipertexto" (Barbosa, 1998).

A estas novas modalidades de criação literária e artística, introduzidas por Pedro Barbosa, estão associados conceitos base que os visam alicerçar, nomeadamente a Literatura Gerada por Computador, a Infoliteratura ou a Ciberliteratura. Estes termos designam um procedimento criativo novo, nascido no berço da Computação, em que o computador é utilizado para a manipulação de signo verbais — assim como para a transmissão de informação (Barbosa, 1998).

A abordagem de Katherine Hayles em *Electronic Literature: New Horizons* (Hayles, 2008) segue um raciocínio similar a Lev Manovich em *The Language of New Media* aludindo às relações entre *software* e cultura. Hayles analisa formas digitais com fim a compreender a especificidade

material e cultural do digital. Perante um contexto sócio-cultural mais amplo da intermediação pessoa-computador, a literatura electrónica torna-se num domínio singular para a observação das mudanças em curso no domínio das tecnologias de produção. O seu enfoque em práticas literárias digitais é parte da sua reflexão crítica sobre um conjunto mais amplo de transformações culturais proporcionadas pela ubiquidade dos computadores.

A segunda geração da literatura electrónica, ou *modernismo digital* (Pressman, 2014), rege-se por obras de arte com um nível de sofisticação maior pela consequência da evolução do paradigma. Dos artefactos categorizados de segunda geração, há paradoxalmente o uso de *software* que facilita e permite a criação de obras multimédia pela agregação de múltiplos tipos de objectos. A título de exemplo, o Adobe Shockwave e Adobe Flash: o primeiro permite a incorporação de trabalhos na Web, implica a existência de um executável para visualização local; o segundo permite a incorporação em *sites* na Web, tendo despoletando um grande interesse — continua a ser usado mas tem vindo a cair, gradualmente, na obsolescência, pela contínua proliferação de dispositivos móveis que não permitem a leitura de Flash e pelo fardo pesado que a maioria dos seus ficheiros implicam que os sistemas carreguem.

Destarte, David Berry em *Understanding Digital Humanities* (Berry, 2012) propõe a existência de uma terceira geração. À medida que as obras de arte, neste caso de literatura electrónica, se complexificam para tirar partido das capacidades da Web, há cada vez mais uma aproximação dos estudos computacionais de *software*, que requer um cuidado especial, uma indagação própria e uma inclusão obrigatória no estudo dos média. Berry incrementa a necessidade de examinar o campo complexo da cultura visto pela tecnologia digital. Para compreendermos a cultura contemporânea dos artefactos nascidos no digital temos que nos focar na programação computacional que está intrínseca em todos os aspectos da cultura e da memória — incluindo a reflexão referente à presença do código (Berry, 2012). Perante esta ordem de argumentos, as tecnologias Web (por exemplo: JavaScript, HTML5, CSS3) possibilitam a produção de obras multimédia em que o seu tamanho é significamente mais pequeno comparativamente a obras em Adobe Flash, que não são compatíveis com a maioria dos dispositivos electrónicos móveis nos dias de hoje.

Deste modo, estamos perante uma conjuntura cultural de dimensão computacional. A importância inegável de que temos que perceber e engajar com a programação. "O código computacional funciona como um índex da cultura digital" (Berry, 2012).

De acordo com (Pressman, 2014), "a literatura electrónica é computacional e procedimental: depende das operações da máquina para atingir os efeitos estéticos e funcionais desejados pelo utilizador". Este paradigma emerge paradoxalmente com o advento da Computação, implicando uma multitude de traduções por intermédio de linguagens de programação, plataformas e redes — resultando da mostra, no monitor, do produto final que está constantemente de mãos dadas com processos algorítmicos, com o *hardware*, com o *software* e com a compatibilidade de *browsers*.

Na obra *Reading Moving Letters: Digital Literature in Research and Teaching* (Simanowski, 2010), Roberto Simanowski realiza uma reflexão extensa para definir literatura electrónica. Serve de especial relevo referir que a condição da Computação digital não é atingida pelo simples facto de ser criada num computador. A condição da criação de objectos no seio digital comporta uma

panóplia de propriedades concretas que se desdobram na conectividade, multimedialidade, não-linearidade, performatividade e transformabilidade. Não obstante muitos textos digitais poderem migrar para livros, a "verdadeira" literatura electrónica não consegue viver sem os média digitais. Por definição, a literatura electrónica tem que ir para além da inserção de palavras — requer um uso estético das ferramentas dos média digitais. Na *e-lit*, a Computação é essencial, não só para texto mas como para qualquer tipo em particular de artefacto físico, mas também para as propriedades literárias específicas do texto. Se as ferramentas da tecnologia digital são essenciais para as propriedades literárias do texto, estas enfraquecem o estatuto dominante do texto. Paralelamente, minar a prevalência do texto como uma máxima linguística conduz à debilitação da natureza digital da literatura electrónica.

Como explica Simanowski, o texto consiste numa aglomeração de letras alfabéticas, a literatura foi sempre o resultado de uma codificação digital. Se a produção de letras é baseada num código binário, como paradigma operacional dos média digitais, a literatura torna-se electrónica num duplo sentido. Porventura, caso aceitemos a premissa de que a literatura electrónica usa tecnologia digital para ser mais do que um texto e se subscrevermos da ideia de que a escrita da literatura electrónica excede a escrita do texto e inclui uma geração de elementos visuais, sónicos e performativos — este raciocínio levar-nos-ia a aferir que a literatura nos média digitais só consiste de letras como unidades digitais. Concluiríamos que não estamos perante literatura electrónica quando as ferramentas específicas dos média digitais não são aplicadas (Simanowski, 2010).

2.3.0.3 Exemplos de Obras de Literatura Electrónica

A obra *88 Constellations for Wittgenstein* por David Clark de 2006 e desenvolvido, em Adobe Flash constitui um bom exemplo deste novo universo. Clark, e a sua equipa, desenvolveram em Flash um conjunto de 88 narrativas digitais ligadas entre si, que expõem factos, personalidades, acontecimentos e argumentos relacionados com a vida de Ludwig Wittgenstein e a conjuntura sócio-económica em que viveu. As narrativas regem-se pelo princípio da obra aberta, termo cunhado por Eco, por terem sido escritas sempre com uma base argumentativa para fomentar discussão e estabelecer novas referências (Clark, 2006).

Do mesmo ano e com recurso ao Flash, serve de referência o trabalho de *e-lit* de Alison Clifford intitulado *The Sweet Old Etcetera* que se situa no género de poesia experimental envolto em música, gráficos, animação, programação e na interactividade do utilizador para atingir caminhos diferentes na narrativa (Clifford, 2006).

O jogo *Façade*, desenvolvido por Michael Mateas e Andrew Stern, é um dos artefactos de literatura electrónica de maior relevo pela inteligência artificial que lhe está inerente — primando pela sua interactividade e ambiente tridimensionalmente. A trama é simples: um casal vive um tumultuoso casamento e o objectivo do jogador é romper o seu laço afectivo (Mateas and Stern, 2005).

O ambiente interactivo e tridimensional de poesia experimental *Poemas do meio do caminho* foi desenvolvido por Rui Torres, em Adobe Flash juntamente com ActionScript. Esta obra permite ao utilizador alterar dinamicamente a sintaxe original dos poemas que residem num conjunto

combinatório de textos programados (Torres, 2009).

2.3.0.4 Conclusão

A literatura electrónica tem vindo a transformar as convenções estabelecidas pela literatura tradicional com a missão de integrar a parafernália de ferramentas computacionais na produção de literatura em moldes impossíveis de replicar no meio impresso. Com a primeira vaga, as obras comportavam essencialmente o uso do hipertexto. Com a segunda geração, evidenciamos a emersão de *software* que permite a fusão de múltiplos tipos de objectos — proporcionando a concepção de artefactos multimédia, com alguma interactividade mas com limitações na sua disseminação. A terceira e corrente geração comporta um vasto número de ferramentas que além de permitirem a interactividade, permitem uma agregação fácil de objectos — o *software* é excluído como principal meio de concepção da obra e as linguagens de programação de alto nível assumem o ambiente de produção.

2.4 Pedro Barbosa

Esta secção visa afunilar o conceito de literatura electrónica com base nos trabalhos de Pedro Barbosa, um dos pioneiros em Portugal desta nova corrente artística. As obras de Pedro Barbosa servem de especial importância para compreendermos a própria disseminação do paradigma da literatura electrónica.

2.4.1 Introdução

Pedro Barbosa, natural do Porto, é licenciado em Letras (especialidade: Filosofia Romântica) pela Universidade de Coimbra e doutorado em Ciências da Comunicação (especialidade: Semiótica) pela Universidade Nova de Lisboa. Barbosa foi docente e investigador em múltiplas universidades do país e estrangeiro: Universidade do Porto, Universidade de Paris X, Universidade de Siena, Universidade Louis Pasteur, Escola Superior de Música e Artes do Espectáculo e Universidade Fernando Pessoa. As suas obras de literatura electrónica estão ligadas à literatura gerada por computador desde 1976. A sua obra é pioneira a nível nacional e internacional no domínio da utilização do computador para geração textual". Correntemente, Pedro Barbosa, tem investigado as potenciais implementações de hipermédia e de geração textual em contextos dramatúrgicos e teatrais.

2.4.2 A Literatura Gerada por Computador

Uma das obras de maior relevo de Pedro Barbosa é o *Sintext* — um gerador automático de textos. Este programa foi desenvolvido em 1996 por Pedro Barbosa e Abílio Cavalheiro, tendo sido posteriormente reinterpretado para versão Java por José Manuel Torres, podendo ser considerado "o primeiro gerador dotado do uso de regras gerativas de informação e não instruções no seu programa" (Bootz and Funkhouser, 2014) — permitido o distanciamento entre algoritmos

que geram material textual e algoritmos que apresentam este material no ecrã. O *Sintext* possui um aglomerado de textos organizados em etiquetas que, por sua vez, comportam múltiplas regras combinatórias que na geração textual de versos soltos, permite uma coerência na sintaxe que é refeita cada vez que regeramos, aleatoriamente, os poemas .

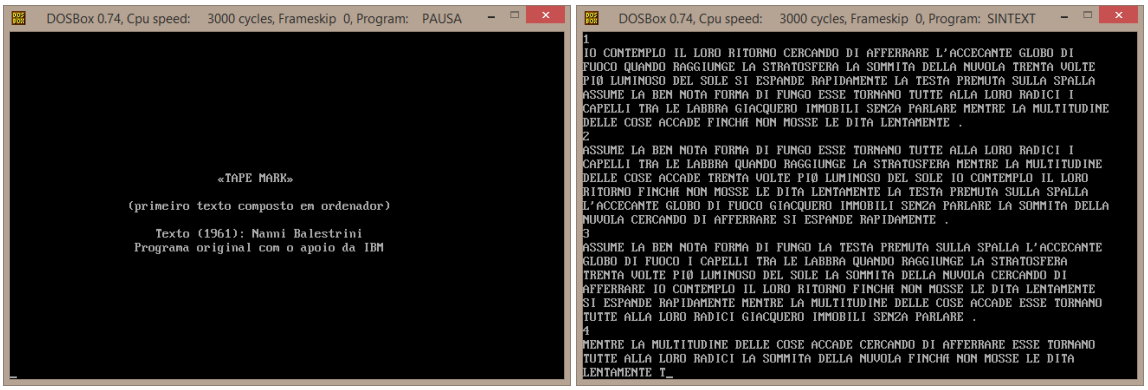
A geração combinatória de textos remonta aos séc. XV e XVI pelos artefactos dos percursores Jean Meschinot e Julius Scalinge, tendo sido implementada na literatura electrónica em 1959 com *Stochastische Texte* de Theo Lutz que fez uso dos textos de *The Castle*, obra de Frank Kafka (Bootz and Funkhouser, 2014). Lutz, um estudante alemão de informática da Universidade Tecnológica de Estugarda, entrou em contacto com processadores electrónicos de dados, controlados por um programa, que por sua vez satisfaziam os objectivos de investigadores em matemática aplicada e Computação — as suas possíveis aplicações eram ilimitadas; não obstante da maioria dos investigadores associarem o uso de processadores electrónicos de dados para a resolução de problemas numéricos, tal limitação era inexistente. Deste modo, Lutz explica que "há a necessidade de aprender a programar um processador electrónico de dados e entender a natureza das suas estruturas" (Lutz, 2005) — cabe ao programador interpretar essas estruturas e adequá-las aos seus objectivos. Dado o contexto, Lutz fez uso de um sistema denominado por ZUSE Z 22 do centro de informática da Universidade Tecnológica de Estugarda que gerava textos estocásticos[53]. Deste modo, a tarefa do programa residia na produção automática de textos estocásticos. Lutz usou a obra de Castle com o programa para gerar uma panóplia de frases aleatoriamente determinadas (Lutz, 2005).

O autor italiano Nanni Balestrini é reconhecido no seio da comunidade de literatura electrónica pela sua obra Tape Mark I[54] de 1961, afirmando-se como um dos primeiros a relaizar experiências de poesia gerada por computador. Balestrini fez uso dos computadores IBM de modelos 7070 e 1401 do Centro Elettronico della Cassa di Risparmio localizado na Província de Lombardo em Milão para colocarem as suas ideias em prática com o auxílio do programador da IBM Alberto Noris. Perante este contexto de indagação pelas novas possibilidades de criação artística que o computador permitia, Balestrini combinou vários elementos linguísticos num programa pré-determinado. Contrariamente a outras experimentações no campo da cibernética que tomam com um dos objectivos replicar ou automatizar procedimentos humanos, Balestrini faz alusão à limitação que impôs nesta experiência, afirmando que a sua obra é somente um exemplo da capacidade de um aparelho electrónico tratar de operações complexas, ligadas a técnicas poéticas, de um modo célere (Zielinski, 2005).

Três décadas depois, Pedro Barbosa reproduz a obra de Balestrini no seu programa Sintext, arquitectado em conjunto com Abílio Cavalheiro na linguagem C++, entre 1993 e 1996 (ver Figuras 2.6a).

Já na década de 1980, Silvestre Pestana, um artista visual e poeta experimental, depois de regressado do exílio na Suécia depois da Revolução dos Cravos em Portugal em 25 de Abril de 1974, trouxe consigo diversas influências do foro da fotografia, vídeo, performance, e suportes

[53] Textos estocásticos: frases em que as palavras são aleatoriamente determinadas (Lutz, 2005).
[54] Tape Mark I: http://www.in-vacua.com/cgi-bin/tapemark.pl



(a) Nota introdutória a poema de Nanni Balestrini (b) Poema a ser gerada por Sintext em C++

Figura 2.6: Poema de Nanni Balestrini no Sintext de Pedro Barbosa

informáticos. Poucos anos depois, na década de 80, o mundo assistiu à introdução de computadores pessoais (PCs), e com isto a uma segunda fase de produção literária por computador. Segundo (Seiça, 2015): "Enquanto que a primeira fase criativa de literatura electrónica esteve ligada aos computadores de grande porte, acessíveis apenas em ambientes institucionais, o contexto em que Silvestre Pestana criou seus primeiros poemas de computador foi totalmente diferente, — caracterizado por um novo movimento a que Pedro Barbosa ironicamente intitula de «poesia doméstica»" — aludindo à possibilidade de criação de poesia por computador no seio de suas casas. Como o auxílio dos computadores Sinclair ZX-81 e Sinclair ZX Spectrum, Silvestre Pestana programou, em BASIC (ver Secção 2.2.4), três poemas respectivamente dedicados a Henri Chopin, E.M. de Melo e Castro e Julian Beck, em que daí resultou nas séries *Computer Poetry* (1981-83) (Seiça, 2015).

Há décadas que o espólio literário de bpNichol tem atraído o interesse de críticos literários que se debruçam no estudo e compreensão dos seus métodos na produção de obras literárias. O género poético de bpNichol assenta numa fusão entre poesia tradicional e princípios matemáticos que se traduzem em desafios para os críticos de literatura analisarem e replicarem os seus artefactos. Algumas das suas obras mais relevantes são *First Screening* (ver Secção 2.2.4) e *Probable Systems* que contêm jogos matemáticos denominados por *alphametrics* e *cryptarithms*. A matemática fora usada por bpNichols para propulsionar a criação artística para outros níveis de complexidade, e com isso trazer uma nova forma de ler poemas (Brown et al., 2009).

A geração combinatória faz uso de estruturas linguísticas de frases existentes para as permutar com enunciados alternativos, endereçando-as a agrupamentos da língua pela mudança e deslocação de textos de acordo com as normas pré-estabelecidas. Os procedimentos combinatórios só podem gerar textos pequenos porque os algoritmos só lidam com a sintaxe e não com o significado do conteúdo de uma obra literária. Como explica (Bootz and Funkhouser, 2014), na geração textual "são frequentemente usados para produzir versos soltos, ou brancos".

O conceito de geração de poesia digital teceu os seus inícios em 1965 por Louis Couffignal, um dos palestrantes das *Conferências Internacionais de Genebra*. Couffignal "distribuiu dois textos: um automático — produzido por um cérebro electrónico de nome Calliope; o outro de produção

humana, escrito pelo poeta francês Paul Éluard" (Barbosa, 1977).

A Fernando Namora, também presente, foi-lhe colocado o desafio de decifrar qual dos poemas tinha sido produzido por uma máquina. Paralelamente ao problema colocado, pairava no ar a ideia, e o mesmo tempo o medo, de a máquina poder conceber um poema com uma qualidade semelhante à de um ser humano. A conclusão retirada pela maioria foi que o poema electrónico, comparativamente ao poema do escritor francês, era menos poético; mas"um bom computador supera insofismavelmente qualquer cérebro humano em ausência de cansaço, em tempo e rendimento de trabalho, em rapidez de cálculo ou em velocidade de escrita" (Barbosa, 1977). Destes argumentos nasce a possibilidade da substituição do homem pela máquina, neste caso, na produção textual. O desenvolvimento algorítmico de um sistema que permite tal produção com base em determinadas instruções fomentadas pelo humano, esteve nas origens do texto produzido pelo cérebro Calliope. Couffignal confessou ter introduzido 300 palavras comuns e usuais na literatura, na memória do computador. Conclusivamente, a máquina deu à luz um poema poeticamente enviesado à priori pelo seu programador. Pedro Barbosa refere adicionalmente que Couffignal tinha introduzido regras de sintaxe restritas para que a máquina seguisse um determinada linearidade no interior da aleatoriedade e que, em consequência, as frases produzidas fizessem sentido e estivessem sintacticamente correctas. Segundo (Barbosa, 1977): "Enquanto o poeta tem capacidade para desrespeitar criativamente as regras da gramática, libertar-se delas, superá-las — e quantas vezes daí não nasce justamente a dimensão poética —, a máquina é-lhes rigorosamente fiel por virtude da programação".

Pedro Barbosa estabelece dois caminhos para a criação automática de textos: o método combinatório e o método aleatório. Estes métodos obedecem a um conjunto de processos inerentes à criação do algoritmo que comportem as capacidades de prolongar, desenvolver, ou criar através das instruções do programador, como são, a título de exemplo, as obras Permuta e Texal.

O método combinatório está inerente à produção de poesia pelo computador através da gratuidade, ou seja que advém naturalmente e espontaneamente sem justificação, e pelo maneirismo: o estilo pessoal determinado pelo autor.

No método aleatório, o computador gera resultados sem critério através da informação que lhe é fornecida, caso o autor estabeleça regras perante a aleatoriedade. Como explica (Barbosa, 1977), "a valorização do aleatório e do caótico na mensagem estética decorre directamente das recentes descobertas da física quântica e da teoria da probabilística, e de todas as especulações tecidas em torno do famoso *princípio da indeterminação* de Heisemberg".

Destes métodos, emerge o modelo estruturalista, inerente à análise da narrativa, que mediante um espectro de dados colocados numa linha espácio-temporal e atribuídos a um conjunto restrito de agentes se possa gerar um processo combinatório que dará resultado a uma multitude de textos possíveis (Barbosa, 1980).

Obras de Pedro Barbosa

As primeiras obras e experimentações em literatura electrónica, levadas a cabo, em 1976, por Pedro Barbosa em parceria com o Eng. Azevedo Machado, foram escritas em Fortran (ver

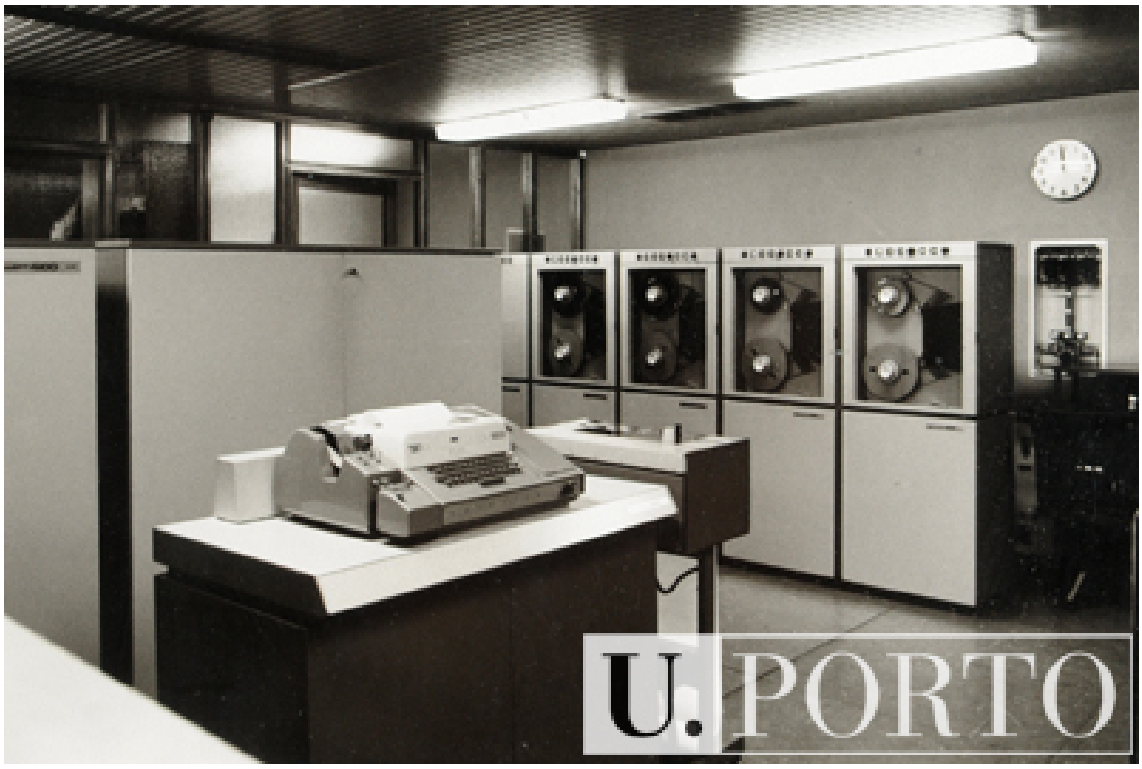


Figura 2.7: LACA da FCUP

Secção 2.2.2) e Algol60 (ver Secção 2.2.3) no Laboratório de Cálculo Automático (LACA) da Faculdade de Ciências da Universidade do Porto (FCUP) que além de permitir o uso de Fortran, mesmissimamente contemplava a programação em Algol60, Neat e com limitação, a Basic, BCPL e afins (Barbosa, 1977). O LACA (ver Figura 2.7) existiu entre 1968 e 1980 até que surgiu o Centro de Informática da Universidade do Porto (CIUP), em 1980, que consigo integrou o espaço e material pertencente ao extinto LACA. Destas experiências surgiram os programas Permuta e Texal. Antes do nascimento das duas obras de literatura electrónica, Barbosa viu-se confrontada com a necessidade de encontrar um algoritmo que permitisse o tratamento de um dado conjunto de elementos linguísticos (letras, vocábulos, frases) para posteriormente ser possível produzir automaticamente textos literalmente plausíveis e assim corresponder à ideia primordial do projecto. Posteriormente ao planeamento e definição do problema, a solução mais viável traduzia-se na "preparação do computador para um tratamento de base permutacional (programa) a actuar sobre um conjunto finito de elementos linguísticos materiais (reportório)" (Barbosa, 1977). Optando-se, assim, por um método combinatório que desse azo à geração de textos poéticos mediante certas restrições ou parâmetros.

No livro *A Literatura Cibernética 2: Um sintetizador de narrativas* (Barbosa, 1980) é demonstrada a arquitectura da segunda versão de Texal que contempla "um conjunto de "textos", separados por "lacunas" para cada uma das quais se fornece uma lista de elementos" (Barbosa, 1980).

"Numerosas têm sido as tentativas esparsas de utilização criativa do computador para

> gerar textos de cariz literário. Muitas vezes não passaram de experiências pontuais, mais ou menos individualizadas e sem carácter sistemático." (Barbosa, 1996a)

Mais tarde, em 1988, a edição de *Máquinas Pensantes (Aforismos gerados por computador)* contempla o código fonte dos poemas *Acaso*, *Aforismos A e B*, e *Re-Text* criados com a linguagem BASIC (ver Secção 2.2.4) (Barbosa, 1988).

No livro *A Teoria do Homem Sentado* de 1996 é contemplado o programa mais relevante da obra de Pedro Barbosa, o Sintext, que vem posteriormente sendo adaptado para outras linguagens; mas na altura fora escrito em C++ (ver Secção 2.2.5) por Abílio Cavalheiro, ex Professor da FEUP, e disseminado em disquete para ser lido no SO MS-DOS. Com a inserção de algumas instruções na linha de comandos é possível seleccionar os poemas que o utilizador pretende ver gerado, e algumas operações podem ser executadas pelo premir das seguintes teclas: '+' para aumentar a velocidade a que o poema é gerado; '-' para retardar a movimento a que o poema escrito; 't' para indicar o valor actual temporizado; 's' seguido de 'x' para interromper o processo; 'a' para ver comandos; 's' para parar/retomar o processo. Nesta obra há o reforço do conceito de obra infinita; ou, nos termos de Umberto Eco, de obra aberta. Segundo Allain Vuillemin, citado por Pedro Barbosa: "uma obra que existe em múltiplas formas, sob estados diferentes, uma obra que está ao mesmo tempo em todo lado e em lado nenhum" (Barbosa and Cavalheiro, 1996). Uma nova forma de ler o texto continua a ser explorada através do auxílio do computador que permite um só texto ser gerado ínfimas vezes sem perder a sua coerência literária. A materialidade do texto é perdida e o texto passa a residir no seio virtual — deste modo, cunhado por Pedro Barbosa, como sendo um "texto virtual". A despeito disto, Pedro Barbosa faz alusão à importância de definir o caráter do livro como sendo de génese electrónica; não por ser distribuído em formato de disquete mas porque quebra os padrões do livro tradicional. Como explica (Barbosa, 1996b): "O texto virtual implica o inacabamento e a multiplicadidade infinita dos textos a *gerar* pelo programa; os textos não existem no suporte magnético enquanto textos, e portanto não detêm um sentido *a priori*, os textos apenas existem no computador em estado *potencial*, em estado *latente*, em estado de *projecto*, em estado de *programa*". Nesta linha de raciocínio, o "texto virtual" é uma estrutura literária que depende de um sistema computacional para existir.

Conclusivamente, o *modus operandi* comunicacional da literatura sofre uma ruptura dicotómica: no meio da criação e mesmamente no seio da recepção. Na leitura submerge a interactividade do leitor na co-criação do texto final (Barbosa, 1988).

No primeiro ano do século XXI surgiria *O Motor Textual* que subscrevia os mesmos contornos de *A Teoria do Homem Sentado*; contudo em formato CD-ROM e numa versão do Sintext arquitectada em Java (ver Secção 2.2.6) por José Manuel Torres. Nesta versão, a despeito de constarem instruções, mesmamente presentes na edição anterior, para o utilizador conseguir aceder e interagir com a obra de arte, constam três textos gerativos: Teoria do Homem Sentado, Balada de Portugal e Didáctica. Contrariamente à versão e programas precedentes, esta é provida de uma interface (ver Figura 2.8) com botões que proporcionam ao utilizador tanto uma maior facilidade em executar os textos assim como mais funcionalidades: aumento ou diminuição da velocidade de escrita, executar um texto infinitamente sem a acção do utilizador, limpar a janela de visionamento,

Figura 2.8: Sintext em Java

entre outros. A linha de comandos é substituída por uma interface intuitiva que proporciona uma experiência de utilização mais rica (Barbosa, 2001).

Posteriormente, a versão do Sintext em Java fora embebida num sítio na Web para permitir a qualquer um com acesso à Internet, o acesso ao programa e, consequentemente, às obras. Contudo, o lugar do Java não reside na Web; apesar de esforços por parte da linguagem, é gradualmente mais difícil correr uma aplicação Java na Web sem encontrar demais problemas e restrições proporcionadas por *browsers*. Houve a necessidade de adaptar o código do programa Sintext para uma linguagem ubíqua entre dispositivos, e desenhada para a Web, como é o caso de JavaScript (ver Secção 2.2.7). Para tal, Nuno Ferreira, em parecia com Rui Torres, escreveu um *script* em 2012, intitulado Poemario.js, que permite reproduzir o mesmo que a versão antecedente do Sintext. Em 2014, foi submetida à ELO do M.I.T.[55], e posteriormente aceite para a terceira colecção de trabalhos de literatura electrónica, uma página Web que além de acoplar o *script*, fora devidamente estruturada com HTML5 e estilizada com CSS3 por Carlos Amaral (ver Figura2.9).

[55] Submissão de Sintext em JavaScript à ELO em: http://po-ex.net/pedrobarbosa/PB_ELC3.html

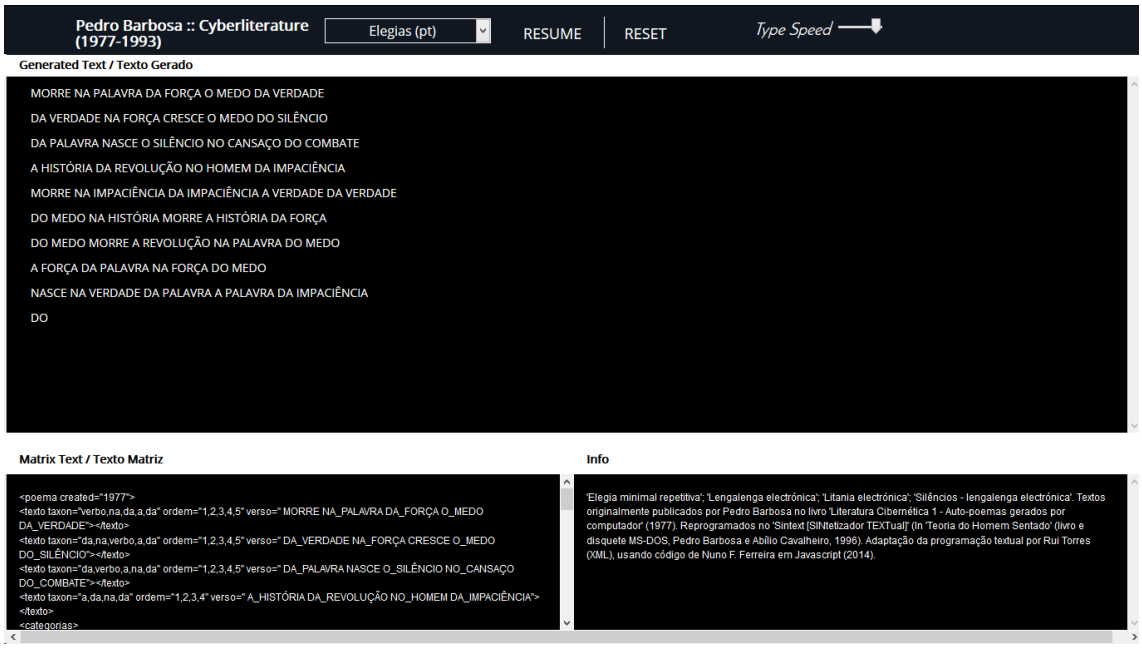


Figura 2.9: Sintext em JavaScript

2.4.3 Conclusão

Pedro Barbosa tem vindo a desenvolver experiências no âmbito da literatura electrónica, mais concretamente de literatura gerada por computador, desde 1976. Inicialmente com famosos cartões (ou fitas) perfurados da IBM com Fortran, atravessando Algol60, BASIC, C++, Java e, nos dias correntes, com JavaScript. O uso do computador para geração textual por parte de Pedro Barbosa foi uma das primeiras experiências com literatura cibernética.

2.5 Edições Críticas Digitais

A transmissão de conhecimento de geração para geração esteve constantemente ligada à natureza da humanidade e da sobrevivência das civilizações. Inicialmente, a transmissão de conhecimento era feita oralmente, a memória do humano funcionava como um arquivo. Posteriormente surgiu a possibilidade de armazenar fisicamente a informação com os pergaminhos, manuscritos, papel, gravuras, entre outros — actividades que requeriam um trabalho editorial. Hoje, com a conjuntura tecnológica em que vivemos, é nos possível armazenar informação digitalmente.

2.5.1 Introdução

A civilização ocidental tem-se preocupado com a transmissão e estudo de textos desde os primórdios da literatura, assim como a criação de um sistema de comentário crítico que visasse comparar textos ou versões de textos. Orígenes de Alexandria, autor cristão, criou a *Hexapla*, uma edição crítica que comparava seis versões do *Velho Testamento*. Desde aí, despoletou-se a necessidade e ambição de produzir edições críticas em papel e/ou manuscrito, atravessando

os sistemas de escrita, a tecnologia de impressão e, correntemente, "a codificação digital que possibilitou um conjunto de convenções e procedimentos algorítmicos que facilitam as operações auto-reflexivas e comunicativas do ser humano" (Clivaz and Hamidovic, 2014).

Com os arquivos digitais, vamos para além do repositório físico ou local final de armazenamento de um dado objecto material. No arquivo digital, um objecto continua a adquirir significado em função da disposição dos objectos por parte dos seus utilizadores num constante ciclo de reutilização, recombinação e representação do objecto.

Neste capítulo são cruzados os conceitos *edição crítica digital* e *arquivo digital* que, apesar de serem complementares, assumem contornos gradualmente mais complexos e comportam particularidades distintas na era digital.

2.5.2 As Origens da Edição Crítica e a sua Relevância no Digital

Uma das primeiras edições críticas, de que há conhecimento, remonta à Grécia antiga. Orígenes de Alexandria revolucionou a transmissão de conhecimento ao criar uma edição crítica intitulada *Hexapla*, uma compilação de seis versões do *Velho Testamento*. A primeira versão é a original em Hebreu. A segunda denominava-se *Secunda*, sendo uma transliteração dos textos em Hebreu para caracteres gregos pela autoria de Orígenes de Alexandria. A terceira é uma tradução de Hebreu para Grego por Áquila de Sinope. A quarta foi traduzida por Symmachus para Grego. A quinta, intitulada de Septuaginta, foi traduzida para Grego helenístico por 70 estudantes judeus. A sexta é uma tradução da autoria de Teodócio, um estudante helenístico, que partiu da quinta versão para desenvolver uma edição melhorada. O professor, estudante, pregador e com o *Hexapla*, escritor — Orígenes de Alexandria, recolheu todas as edições para as comparar. De acordo com (Law, 2008), "[a]s motivações do autor nesta edição crítica estavam adjacentes à necessidade de estabelecer uma versão correcta da obra" como uma ferramenta apologética que providenciava aos cristãos um meio de defesa contra as controvérsias com os judeus.

O colectivismo e a curadoria são actividades presentes no desenvolvido de arquivos e edições críticas que envolvem listar, catalogar e criar inventários que estiveram sempre presentes na literatura como um dos mais antigos modelos de estudo e análise literários. Os inventários remontam à Grécia antiga e a escritores como Hesíodo, Ésquilo, Sófocles e Eurípides. O primeiro catálogo de bibliografia foi criado por Calímaco, na sua obra de arte intitulada *Pinakes*, inscrito em 120 tabletes que continha todo o conteúdo numerado da biblioteca de Alexandria. Mais tarde, nas obras de Homero há a inclusão de catálogos que combinam o texto e a imagem com o intuito de fornecerem informação compacta e concisa (McGann, 2010).

Com as edições críticas digitais não só temos um arquivo digital mas temos interpretações literárias. Julia Flanders cita os conceitos de crítica baixa e crítica alta — cunhados por Ray Siemens (Flanders, 2005). O primeiro preocupa-se em estabelecer o texto como um objecto de estudo com a edição crítica; o segundo encarrega-se de usar os resultados como base para interpretações. Nesta ordem de ideias os responsáveis pelo *criticismo baixo* são os editores; por outro lado, os interpretadores que estavam associados ao *criticismo alto* eram vistos como "sumos sacerdotes do significado" (Flanders, 2005).

Anteriormente, os termos *arquivo digital* e *edição crítica* significavam o mesmo, mas com o advento das ferramentas, bases de dados e artefactos multimédia, o arquivo vem agora representar algo totalmente diferente. Segundo a definição de (Flanders, 2005), "a edição crítica digital poderá ser compreendida como um arquivo de duas camadas editoriais: uma operando como os factos estabelecidos e imutáveis do texto e a outra funcionando como o domínio mutável da perspectiva e interpretação humana".

Com a advento dos computadores é introduzido o conceito de base de dados como forma narrativa. Os artefactos dos novos média "não contam histórias; não possuem um princípio e um fim" (Manovich, 2001) — são colecções de objectos individuais, espalhados por bases de dados. Manovich define *base de dados* por "uma colecção estruturada de informação" (Manovich, 2001). Há cinco tipos de bases de dados (hierárquica; em rede; relacional; orientada a objectos; relacional-objecto (NoSQL), cada uma possuidora de modelos díspares de organização da informação. O modelo hierárquico funciona em árvore. Os orientados a objectos guardam estruturas complexas de dados denominados por objectos que estão, por sua vez, distribuídos numa hierarquia de classes. A concepção de uma base de dados digital assenta num conjunto de objectivos: 1) procura célere de informação; 2) retorno de dados por um computador; 3) apresentação da selecção feita pelo utilizador. Podemos aferir que a experiência do utilizador na retenção de informação através de bases de dados digitais é distinta das narrativas na leitura de um livro ou no visionamento de um filme. Nos novos média as bases de dados são armazenadas em CD-ROMs, DVDs, discos Blu-ray, USB-pens, contendo enciclopédias, colecções, dicionários, imagens, vídeos e afins. Algumas dessas bases de dados podem assumir contornos multimédia em que imergem o utilizador numa experiência virtual que comporta imagem estática e em movimento, animação, narrativa, áudio, entre outros (Manovich, 2001). Todavia, os novos média não são sinónimos de base de dados. Os vídeo-jogos, na sua maioria, não assentam nessa categoria — apesar de possuírem uma edificação que obedece a um modelo de base de dados, os utilizadores raramente têm acesso a um sistema de base de dados. Os objectivos normalmente regem-se por cumprir os objectivos do jogo, obter a pontuação mais alta, competir entre jogadores, subir de nível, entre outros. Não obstante desta premissa que está presente na maioria dos vídeo jogos, o *Grand Theft Auto V* possui bases de dados que permitem o acesso a inúmeros objectos e informação que fazem parte da experiência do utilizador no jogo — e este facto é gradualmente mais evidente. O vídeo-jogo, nos dias correntes, vai além de uma simples narrativa linear, assumindo ramificações extraordinárias em que o utilizador é exposto a um universo de alternativas — especialmente quando conectado à rede em que há uma troca e distribuição contínua de informação por parte da empresa e seus jogadores espalhados pelo mundo.

Transversalmente à literatura electrónica (ver Secção 2.3), as edições críticas digitais são edições de artefactos digitais que, por isso mesmo, despoletam outros e novos problemas relacionadas com a sua representatividade, marcada pelas bases de dados, pela transformação, pela variabilidade que lhe está inerente e, consequentemente pela obsolescência tecnológica (ver Secção 2.1.6) — mais do que exemplos de edições críticas de obras em papel, posteriormente preservadas e representadas pela digitalização.

2.5.3 Exemplos de Edições Críticas Digitais

Nesta subsecção são descritos exemplos de edições críticas digitais e de arquivos digitais maioritariamente desenvolvidos em contexto universitário para fins de investigação e ensino.

O *Arquivo Digital PO.EX* é um sítio na Web em Po-Ex (dot) Net, desenvolvido por Rui Torres em 2005, que comporta um vasto espólio de obras de poesia experimental portuguesa, digitalmente curadas (ver Secção 2.1.5), num arquivo digital. Segundo (Torres, 2005), o objectivo deste projecto foi recolher, classificar, digitalizar e reproduzir em formato electrónico, a produção da poesia concreta e visual portuguesa associada ao Movimento da Poesia Experimental dos anos 60 (conhecido como PO.EX), com vista à produção de um CD-ROM de divulgação da mesma. Considerou-se para esse efeito como espólio da PO.EX os Cadernos, Suplementos, objectos, catálogos e panfletos".

Num projecto colaborativo, partindo da iniciativa de Gregory Cane, entre investigadores e estudantes de várias instituições, datado a 1992, o *Perseus Project* é uma plataforma extensível, desenhada no *HyperCard* da Apple, com conteúdo hipermédia sobre a cultura antiga grega. O processo de concepção deste arquivo digital remonta a 1985, a sua fase de planeamento, e posterior início de desenvolvimento em 1987. A Yale University Press, em 1992, publica a primeira versão do projecto *Perseus 1.0* em CD-ROM e num disco óptico. Quatro anos mais tarde, em 1996, surgiu a segunda versão desta obra, desta vez "com uma narrativa que proporcionava uma visão geral da cultura antiga grega" (Crane, 1996) que alicerçava o conteúdo de um livro digital de 300 páginas com centenas de hiperligações.

O *Perseus Project* comporta textos gregos originais com traduções em Inglês por mais de 10 autores; uma visão global histórica com hiperligações para textos, mapas e imagens. Comporta ainda um conjunto de ferramentas que auxiliavam estudantes ou investigadores, ao possuírem entradas de léxico com traduções em Inglês; um índex com definições em Inglês; uma base de dados morfológicos para palavras em textos gregos. A panóplia de imagens coloridas é representativa dos objectos e traços arquitectónicos da Grécia antiga, desenhos de edifícios e seus diagramas ligados a um índex com um catálogo e palavras-chave para todos os objectos. Esta plataforma permite a interactividade entre o utilizador que pode guardar, anotar e/ou editar o seu percurso pelo arquivo digital (Marchionini and Crane, 1994).

Um das necessidade que os autores do projecto tiveram, após o lançamento da primeira versão, foi permitir que o *Perseus Project* fosse ubíquo e não estivesse confinado à dependência do *HyperCard*. Para isso foi desenhada uma Aplicação Web (Crane, 2014) em 1995. Porém, na altura era impossível transferir todo o conteúdo da plataforma criada no *software* da Apple para o sítio na Web pela "incapacidade do *HTML 2.0* e do *Netscape 1.0* conseguirem replicar as funcionalidades presentes no *HyperCard*" (Crane, 1996). Conclusivamente, o *Perseus Project* "é uma iniciativa importante porque conseguiu converter o corpus da literatura clássica num modelo digital" (Burdick et al., 2012). A plataforma teve um contínuo progresso e actualmente vai na quarta versão, agora denominada por *Perseus Hopper*.

O artefacto digital *The William Blake Archive* (Eaves et al., 2015), publicado em 1996, foi

desenvolvido por Morris Eaves, Robert Essick e Joseph Viscomi, em colaboração com os investigadores do *Institute for Advanced Technology* da *University of Virginia*. Como explica (Jones, 2006): "O projecto é uma das primeiras edições estudantis em formato electrónico expressamente desenvolvida para publicação na WWW, e permanece um recurso online gratuito". A plataforma comporta uma representação digital de pinturas, livros, desenhos, gravuras, entre outros, do artista de origem inglesa William Blake que viveu entre o séc. XVIII e XIX. As imagens e textos presentes no arquivo foram codificados nos formatos ISO: TIFF e JPEG para imagens, SGML e TEI para textos — adoptando posteriormente o XML e o HTML. Os percursores do *The William Blake Archive* conseguiram criar uma plataforma Web independente, robusta e persistente: contemplando padrões reconhecidos no seio da comunidade e gramáticas técnicas; com enfoque no estudante como público-alvo primário; aberto a utilizadores espalhados pelo mundo. Como explica (Schwartz, 2000): "The William Blake Archive vai além da maioria ao proporcionar uma visita guiada virtual muito bem desenvolvida, uma extensa documentação de ajuda, a história do projecto, uma FAQ, uma bibliografia, e informação em princípios editoriais, os editores, e a tecnologia".

Jerome McGann, desenvolveu de 1993 a 2008 o arquivo digital *The Rossetti Archive* com todas as obras de Dante Gabriel Rossetti. A primeira versão da plataforma fora lançada em 2000, seguindo posteriormente outras três até estar totalmente completa. Esta obra digital partiu da investigação de como "desenvolver e criar ferramentas educacionais, interpretativas assim como editoriais, para investigação e publicação *online*" (McGann, 2010). Do foro editorial, McGann e a sua equipa puderam evidenciar que os modelos tradicionais editoriais de textos eram claramente obsoletos comparativamente ao modelos editoriais digitais. Nesta linha de pensamento, McGann traçou um conjunto de objectivos hipermédia, que visavam a inserção de textos, livros, imagens e áudio. De acordo com (McGann, 2008), "todos os documentos estão codificados para análise estrutural e analítica". Contudo, McGann refere a ausência de interactividade visto que *The Rossetti Archive* comporta a representação de informação que pode ser visualizada e aumentada mas a base em que o arquivo foi fundada não pode ser reconstruida ou reconfigurada. Apesar desta limitação, esta edição crítica alcançou o sucesso de ter a obra completa de Dante Rossetti disponível ao alcance de dois cliques.

O *Project Gemini Online Digital Archive* é mais um dos exemplos a referir de um arquivo digital que possui digitalizações originais das imagens capturadas a voo do *Project Gemini* — o segundo programa de voos espaciais dos EUA. "O arquivo digital possui imagens de 10 missões" (Showstack, 2012) levadas a cabo pela tripulação de 10 elementos.

"Anotar o mundo" é a premissa da edição crítica digital Genius (dot) Com, fundada em 2009 por Mahbod Moghadam, Tom Lehman e Ilan Zechory. A sua origem está na música Rap, sendo o seu nome anterior *Rap Genius*. No curso da sua evolução expandiram as áreas temáticas e incluíram Rock, Literatura, Cinema, Notícias, História, entre outros. É possuidora de uma interface intuitiva e inovadora, todos os utilizadores podem criar publicações e posteriormente anotarem textos com as suas interpretações — podendo inserir todos os tipos de imagens, hiperligações e usar as mais diversas formatações de texto. A plataforma funciona em comunidade, qualquer

membro pode contribuir para um texto ou expor uma interpretação diferente do que fora dada por outro utilizador. Actua como uma base de dados para o conhecimento e "almeja ser a próxima Wikipédia" (Moghadam et al., 2009).

Correntemente há uma colaboração entre o *Israel Antiquities Authority* e a *Google* para digitalizar todos os 940 manuscritos do Mar Morto e disponibilizá-los na Web. O objectivo deste arquivo digital tem como finalidade o fácil acesso a esta informação por parte da comunidade científica (Harris, 2014).

2.5.4 Conclusão

A *Hexapla* de Orígenes de Alexandria vem catapultar o conceito de *arquivo* para um nível totalmente diferente, vem adicionar interpretações ao conteúdo arquivado. Em plena era da Computação, a disseminação de conhecimento nunca antes foi tão simples: a Internet possui um universo de conhecimento maior do que o sábio da Grécia antiga. Deste modo, a edição crítica digital é uma colecção de artefactos electrónicos multimédia que cobrem uma variedade de tópicos estruturalmente organizados. A Genius (dot) Com é provavelmente a edição crítica digital tecnologicamente mais avançada em que permite anotar qualquer Website mais relevantes fazendo uso das múltiplas ferramentas que as bases de dados e a Web 2.0 proporcionam.

Capítulo 3

Especificação da Aplicação Web

Este capítulo contém a descrição dos requisitos e das funcionalidades que a Aplicação Web, intitulada *Pedro Barbosa: Edição Crítica Digital*, deve respeitar para cumprir o conjunto de objectivos que estão intercalados nas fases de início, elaboração e construção do sistema.

3.1 Descrição do Problema

O problema que o trabalho visa resolver rege-se à inexistência de uma edição crítica digital que vise fornecer e aglomerar todo o espectro de obras de literatura electrónica arquitectadas por Pedro Barbosa desde 1976 até ao presente. Para além de um arquivo digital com ligações internas, há a necessidade da implementação de funcionalidades que proporcionem a interactividade e o comentário crítico. Para enquadrar o conjunto de artefactos de literatura electrónica, terá que haver recursos que permitam contextualizar o leitor perante as conjunturas tecnológicas e sócio-económicas que estiveram presentes no paradigma da literatura electrónica. Deste modo, é fulcral a existência de textos estruturados em linhas temporais que expliquem cronologicamente o envolvimento das Humanidades com o advento da computação e, consequentemente, as linguagens de programação e *software* que serviram de ferramentas para a concepção de obras de literatura electrónica.

A Edição Crítica Digital, intitulada *Pedro Barbosa: Edição Crítica Digital*, comporta as obras digitais de Pedro Barbosa e visa operar como um repositório de conteúdos que estão contemplados na dissertação numa edição crítica digital — que além de ser um arquivo digital, comporta interactividade. O objectivo desta Aplicação Web reside na difusão de conhecimento no seio da literatura electrónica, com especial enfoque das obras de Pedro Barbosa. Deste modo, funciona como um arquivo digital com hiperligações internas que aglomera uma multitude de documentação que serve de proveito para investigadores e/ou interessados na área de estudos vigente.

Este sistema está talhado para servir de modelo para futuras implementações de outros autores de literatura electrónica. Não obstante, a sua infraestrutura deverá ainda sofrer alterações que proporcionem uma experiência de utilização superior, a adição de funcionalidades e um *layout* esteticamente mais favorável mediante as necessidades. A informação poderá ser enriquecida pelo

contributo de utilizadores/investigadores que se podem registar e inserir comentários nas páginas que comportam a funcionalidade, além do formulário de contacto que dispensa registo.

3.2 Especificação de Requisitos

A Especificação de Requisitos estabelece, no âmbito da dissertação, uma base de concordância entre o criador e o utilizador no que diz respeito ao que o *software* deve fazer e o que é esperado que faça. Esta etapa compreende uma análise rigorosa dos requisitos antes que o desenho da Aplicação Web seja executado.

O requisito é a especificação do que deve ser implementado — bipartido em requisitos funcionais e não funcionais. Os requisitos são os alicerces de um sistema, "transmitem o que é que um sistema deve fazer mas não como é que o deve fazer" (Arlow, 2005). Deste modo, os requisitos são usados para descrever as funcionalidades envolvidas na gestão de desempenho, documentação e manutenção de um conjunto de requisitos para um sistema.

3.2.1 Actores

Um actor especifica um papel que uma entidade externa adopta quando interagindo directamente com o sistema. De acordo com (Arlow, 2005), um actor "pode representar um papel de utilizador ou um papel desempenhado por outro sistema ou peça de *hardware* que entra em contacto com os limites do sistema". Para a plataforma *Pedro Barbosa: Edição Crítica Digital* a ser alojada no Arquivo Digital de Poesia Experimental Po-Ex (dot) Net[1], consideram-se os actores na Figura 3.1 e na Tabela 3.1.

Tabela 3.1: Tabela de Actores

Identificador	**Descrição**	**Exemplos**
Utilizador	Utilizador não-registado — Visualiza o conteúdo mas não está registado para comentar publicações ou adicionar informação. Pode partilhar as publicações nas redes sociais.	n/a
Visitante	Utilizador não-registado — Pode registar-se ou autenticar-se no sistema.	n/a
Leitor	Utilizador registado — Visualiza o conteúdo e tem a possibilidade de comentar publicações sujeitas a aprovação por um Administrador. Pode partilhar as publicações nas redes sociais.	João
Administrador	Utilizador registado — Pode promover ou despromover utilizadores, além de aprovar ou rejeitar comentários.	Administrador
Proprietário	Utilizador registado — Detém os direitos de autor da Aplicação Web.	Carlos Amaral

[1] Po-Ex: http://po-ex.net/

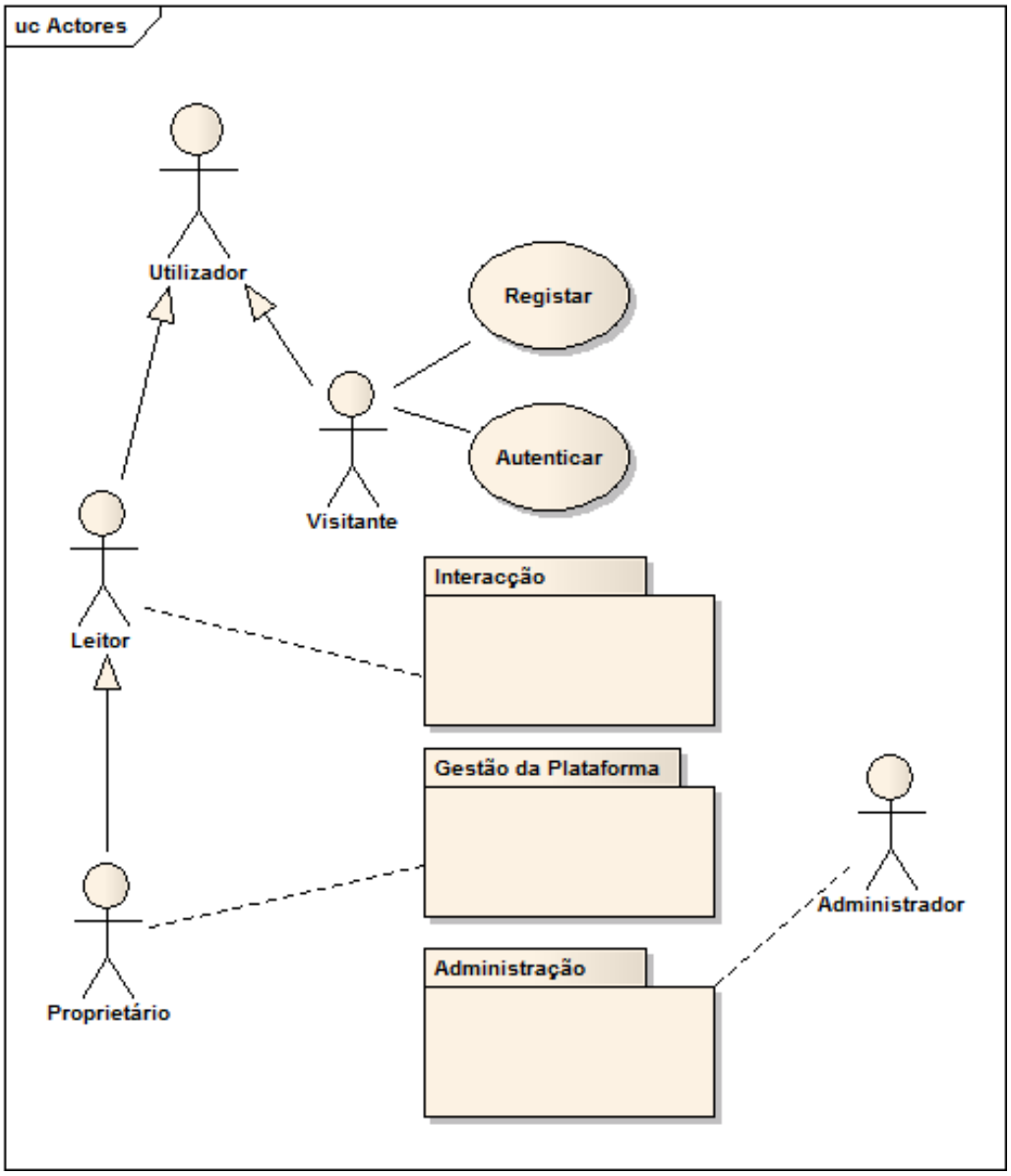


Figura 3.1: Diagrama de Actores e Casos de Utilização

Tabela 3.2: Narrativas de Utilização

Identificador	Nome	Prioridade	Descrição
US01	Autenticação	alta	Como Visitante quero ter a possibilidade de me autenticar no sistema para poder inserir comentários e para aceder a informação privada caso seja Administrador
US02	Registo	alta	Como Visitante quero ter a possibilidade de me registar no sistema para me poder autenticar
US03	Pesquisa	alta	Como Utilizador quero fazer uma pesquisa para obter toda a Informação existente sobre o assunto
US04	Comentário	alta	Como Leitor quero comentar as publicações disponíveis
US05	Perfil	alta	Como Leitor quero aceder ao meu perfil e alterar a informação proporcionada por mim no momento de registo (endereço de e-mail, palavra-chave, etc) e adicionar uma fotografia de perfil
US06	Comentários Pendentes	média	Como Administrador quero aprovar ou rejeitar comentários de Utilizadores
US07	Privilégios	média	Como Administrador quero alterar o tipo de utilizador de Normal para Administrador ou vice-versa, ou eliminar um utilizador
US08	Editar Página	média	Como Administrador quero editar o campos de texto em vistas para alterar o seu conteúdo
US08	Terminar Sessão	alta	Como utilizador quero terminar a minha sessão
US09	Partilhar	média	Como utilizador quero partilhar as páginas no Facebook e Twitter

3.2.2 Cenários de Utilização

Um cenário de utilização especifica uma sequência de acções, incluindo sequências e erros variantes que um sistema pode executar ao interagir com actores externos. Destarte são apresentadas as acções na Tabela 3.2, ou narrativas de utilização, que um actor espera que o sistema execute (Arlow, 2005).

3.2.3 Requisitos Técnicos

Na Tabela 3.3 são apresentados os requisitos técnicos que o sistema tem que possuir para garantir o seu bom funcionamento.

3.2.3.1 Modelo Conceptual do Domínio

Nesta secção é levada a cabo a tarefa de descobrir quais os tipos de entidades que representam o conteúdo e os conceitos, assim como os seus relacionamentos, relevantes para a solução do

Tabela 3.3: Requisitos Técnicos

Identificador	**Nome**	**Descrição**
TR01	Disponibilidade	O sistema deve estar *online* 24/7
TR02	Acessibilidade	O sistema deve garantir a ubiquidade entre dispositivos de diferentes características e resoluções, e *browsers*
TR04	Usabilidade	O sistema deve ser intuitivo e de fácil navegação
TR05	Desempenho	O sistema deve ter tempos de resposta inferiores a 1 segundo para engajar o utilizador
TR06	Tempo de Resposta	A aplicação deve proporcionar um tempo de resposta não superior a 0.8 segundos por cada vista.
TR07	Robustez	A aplicação deve ser capaz de lidar com erros, falha humana ou não, e recuperar sem afectar o utilizador
TR08	Aplicação Web	O sistema deve ser implementado como aplicação Web com páginas dinâmicas (HTML5, CSS3, JavaScript e PHP)
TR09	Portabilidade	O sistema do lado de servidor deve funcionar em várias plataformas (Windows, Linux, Mac OS)
TR10	Base de dados	O sistema deve usar o sistema de gestão de bases de dados PostgreSQL 9.3.5
TR11	Escalabilidade	O sistema deve estar preparado para evoluir em termos de funcionalidade e mediante o aumento do número de utilizadores
TR12	Ética	O sistema deve respeitar os princípios éticos no desenvolvimento de *software* (encriptação de palavras-chave com *hash* assegurando que só o utilizador é que a conhece)
TR13	Segurança	O sistema deve proteger a informação de acessos não autorizados (neste caso, através de *middleware*[2]) através da utilização de um sistema de autenticação e verificação de privilégios

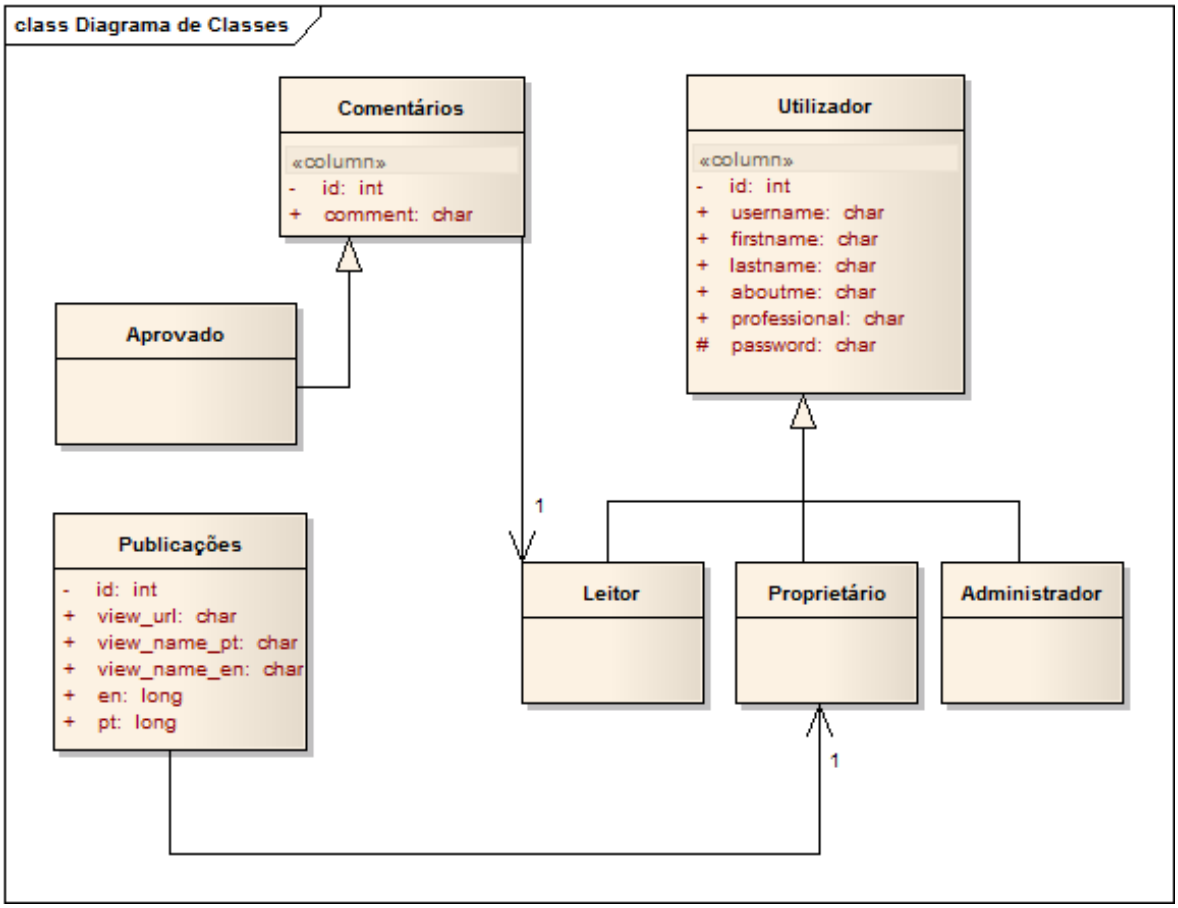


Figura 3.2: Diagrama de Classes

problema (Ambler, 2004). De forma a traçarmos os conceitos centrais do problema, foi desenhado um diagrama de classes UML (ver Figura 3.2) que apresenta as entidades organizacionais e as associações entre elas.

3.3 Perspectivas de Solução

Com o objectivo de proporcionar uma solução ao problema proposto, haverá o desenho conceptual de uma edição crítica digital que funcione como um arquivo das obras de Pedro Barbosa e como um meio para o comentário crítico/interpretativo — contando com um suporte teórico que contextualize o paradigma da literatura electrónica.

A plataforma escolhida para o desenvolvimento desta edição crítica digital é o Laravel que funciona num *Model-view-controller* (MVC), contemplando o PHP e o SQL como linguagens de *back-end* e o HTML5, CSS3, JavaScript, jQuery como linguagens *front-end*. A plataforma será bilingue: Português e Inglês, interactiva e comportará o *grid* do Twitter Bootstrap[3] que permite a ubiquidade entre dispositivos; por exemplo: adaptação a diferentes resoluções e *browsers*.

[3]Twitter Bootstrap: http://getbootstrap.com/

O resultado final deste projecto visa ser um contributo para o campo das Humanidades, procurando disseminar o paradigma da literatura electrónica e as obras de arte de Pedro Barbosa, um dos pioneiros de literatura electrónica.

3.4 Desenho da Arquitectura Lógica

Esta actividade é levada a cabo por um ou mais arquitectos e visa determinar como é que as funcionalidades presentes nos requisitos devem ser implementadas (ver Figura 3.3) (Arlow, 2005). O sistema tem um único ponto de acesso: o *web browser*. O *web browser* serve de meio para demonstrar a informação contida na aplicação aos utilizadores, mediante as regras de negócio implementadas no servidor que acedem e inserem informação no servidor pelo uso de objectos PDO[4] do PHP. Mais se pode aferir que, nas regras de negócio, constam as características referentes aos utilizadores em geral: autenticação, registo, gestão e perfil; não obstante, é relevante referir as opções inerentes às publicações: barra de pesquisa, edição dinâmica de texto, inserção de comentários (submissão pendente a aceitação, por omissão).

3.4.1 Descrição Geral da Arquitectura

Para o desenvolvimento do sistema e solução do problema foi escolhido o Laravel, uma ferramenta em *Model-View-Controller* (MVC) que comporta três tipos de componentes: *model*, *view* e *controller*. O *model* notifica as *views* associadas e controla, com o auxílio dos *controllers*, se houve mudanças no seu estado. Esta notificação permite que a actualização das *views* seja possível e que os *controllers* alterem o conjunto de comandos disponíveis.

3.4.2 Tecnologias na Aplicação

As tecnologias que são apropriadas para a concepção do sistema são: HTML5; CSS3; JavaScript/jQuery; PHP e a ferramenta baseada em PHP — Laravel; PostgreSQL; componentes Web tais como Twitter Bootstrap[5], Font Awesome [6] e Symfony[7].

A linguagem de marcação HTML5 e a de estilização CSS3 proporcionam-nos uma quantidade vasta de ferramentas para o desenvolvimento de uma excelente experiência de utilização. Ambas as linguagens são suportadas pelo *grid* do Twitter Boostrap permitem que a aplicação seja responsiva.

Deste modo, é importante mencionar a relevância de jQuery, biblioteca de JavaScript, para agilizar a manipulação de HTML5 e CSS3 em páginas dinâmicas.

[4] PDO (PHP Data Objects) é um módulo inerente ao paradigma das linguagens de programação orientadas a objectos. Serve para dar coesão à forma como o PHP comunica com as bases de dados relacionais.

[5] O Twitter Boostrap é a uma *framework* para o desenvolvimento Web a nível de *front-end* que comporta componentes Web nas linguagens HTML5, CSS3 e JavaScript — `http://getbootstrap.com/`

[6] Font Awesome proporciona um conjunto vasto de ícones, totalmente flexíveis, esteticamente agradáveis e de fácil implementação — `http://fortawesome.github.io/Font-Awesome/icons/`

[7] Symfony é uma *framework* de PHP com componentes Web — `https://symfony.com/`

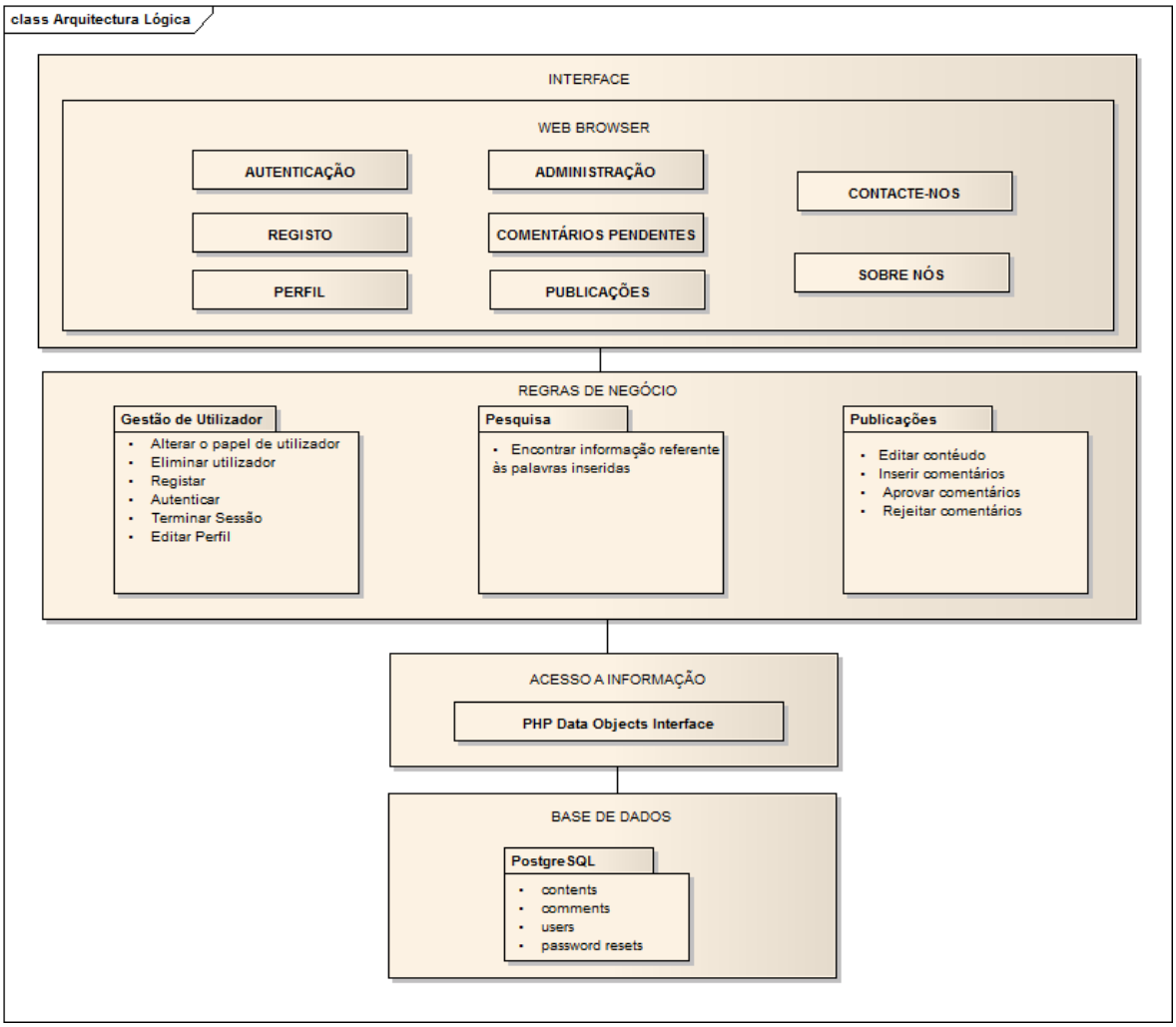


Figura 3.3: Arquitectura Lógica

O PHP foi escolhido em detrimento de ASP.NET para tratar da comunicação entre cliente e servidor; não obstante de possuírem mais ou menos a mesma qualidade de recursos, segurança e extensibilidade, o PHP é *open-source* e proporciona os meios para o cliente escolher uma grande variedade de tipos de servidor (a título de exemplo: permite uma maior facilidade de transferência para servidores online); outras razões depreendem-se do facto de que o ASP.NET faz uso de mais recursos num servidor Web do que o PHP — consequentemente, requer melhores servidores ou um número maior deles. A despeito da experiência em PHP ser pouca, este projecto serve como mais um meio para a aprendizagem de uma linguagem de programação importante na actual conjuntura tecnológica.

No que toca à base de dados, para a gestão de informação, foi escolhida a PostegreSQL por ser umas das bases de dados de modelo relacional mais usada no seio da comunidade *open-source*. A PostgreSQL é compatível com a maioria das *frameworks* e consta em grande parte dos servidores. O uso da base de dados destina-se a guardar informação referente a utilizadores, a informação inserida por utilizadores e ao conteúdo de vistas.

No desenvolvimento de uma experiência de utilização na web 2.0, foi usada a versão 5 do Laravel que nos fornece uma parafernália de recursos incluindo alto nível de segurança, testes unitários, *caching*, a migração do base de dados e capacidade de expansão (ou *Web scale*[8]).

Porquê Laravel?

A popularidade das *frameworks* para o desenvolvimento Web de aplicações têm vindo a ganhar predominância nos últimos 5 anos. Assistimos a uma transição entre escrever código de raiz para *frameworks* que possuem componentes e funcionalidades por omissão. O PHP puro é rejeitado por muitos programadores por ser demasiado confuso e pouco intuitivo; mas no seio das *frameworks* em PHP essa aversão à linguagem foi diminuída. CodeIgniter[9] foi uma das primeiras *frameworks* a se destacar pela sua capacidade de facilitar a criação de funcionalidades em PHP; porém mais tarde a *framework* foi incapaz de se adaptar às actualizações que a linguagem foi sofrendo e de tirar partido das mesmas — implicando problemas de compatibilidade e rupturas no funcionamento de aplicações. Posteriormente, surgiram Symfony e FuelPHP[10] em resposta a este sintoma (Bean, 2015).

Sensivelmente meia década depois é que emergiu Laravel, quando a comunidade de programadores em PHP já tinha ao seu dispor uma parafernália de *frameworks*. O conjunto de objectivos de Laravel não residiam na criação de uma *framework* que contornasse os problemas das anteriores; em contraste, fez uso dos componentes de Symfony para se desenvolver — resultando numa *framework* baseada em componentes. Como argumenta (Bean, 2015), a *framework* escolhida para

[8] Web scale: uma propriedade desejável para um sistema que se traduz pela sua capacidade de se adaptar mediante o número de utilizadores.

[9] CodeIgniter: `http://www.codeigniter.com/`.

[10] FuelPHP: `fuelphp.com`.

Figura 3.4: Página Inicial

o projeto "em vez de providenciar dezenas de bibliotecas inflexíveis, o Laravel proporciona componentes sensíveis e orientados a *drivers* que programadores podem usar para criarem aplicações sem condicionamentos, evidenciados noutras *frameworks* no foro do *layout*".

A versão 3 de Laravel catapultou-o para uma das mais relevantes *frameworks* MVC em PHP. À medida que foi evoluíndo até à versão 5, as *frameworks* já mencionadas, assim como CakePHP e Yii[11] perderam a corrida com Laravel.

3.5 Implementação

Findado o planeamento e a descrição de todas as componentes que estão inerentes à aplicação, serve de especial importância demonstrar o protótipo e suas ramificações.

A Página Inicial, ou *Homepage* (ver Figura 3.4) possui um cabeçalho, presente em todas as páginas, com o logótipo da Aplicação (as iniciais do autor), um ícone que expande um campo de texto ao clique, botões para as Obras de Digitais de Pedro Barbosa, *switch* para alteração de língua, de Português para Inglês e vice-versa e, por último, botões para a Autenticar o utilizador caso esteja registado ou Registar um novo utilizador.

Na página inicial também figura acesso a uma linha temporal (ou *timeline*) que permite expor facilmente ao utilizador quem é Pedro Barbosa e qual o seu trabalho na linha da literatura electrónica e poesia experimental (ver Figura 3.5). A *timeline* permite estratificar um aglomerado de informação num sistema métrico — ajustável em termos de escala temporal (horas, minutos,

[11] YiiFramework: http://www.yiiframework.com/.

Figura 3.5: Linha Temporal das Obras de Pedro Barbosa

segundos, *frames*) (Fuller, 2008). Para a criação da linha temporal digital foi usada uma biblioteca de JavaScript, TimelineJS[12] — paralelamente com JSON[13].

Após a Autenticação e/ou Registo do utilizador é imediatamente possível que este aceda ao seu Perfil (ver Figura 3.6) para aceder às seguintes funcionalidades: adicionar uma fotografia, editar informação, e/ou alterar palavra-chave.

No cabeçalho da interface é possível identificar uma lupa (ver Figura 3.7) que funciona como um botão que, ao premir, despoleta um campo de texto para o utilizador inserir caracteres que estejam, ou não, presentes na base de dados. Após o clicar da tecla ENTER do teclado, há um redireccionamento para uma página que mostra os resultados da pesquisa.

Na página Comentários Pendentes (ver Figura 3.8), somente acessível a Administradores, é possível aceitar ou rejeitar comentários de utilizadores. Num segundo painel, há a mostra dos comentários que foram aprovados e, que por sua vez, se encontram públicos nas vistas e suas respectivas secções.

No painel Privilégios de Administrador (ver Figura 3.9), somente acessível a Administradores, é possível alterar os papéis dos utilizadores, de Normal para Administrador, ou eliminá-los da base de dados.

Por último, serve de especial importância demonstrar a estrutura em que as obras de Pedro Barbosa se regem (ver Figura 3.10). Esta vista está repartida em quatro secções: 1) contextualiza o utilizador perante o artefacto digital que está presente na página ao lhe fornecer dados históricos e tecnológicos sobre o que está para ser demonstrado; 2) demonstra a obra a ser executada mediante

[12] TimelineJS: `http://timeline.knightlab.com/`.

[13] JSON é um formato de dados leve, semi-estruturado e flexível presente em JavaScript que se traduz numa colecção de nomes e valores que são lidos pela biblioteca de JS (Liu et al., 2014).

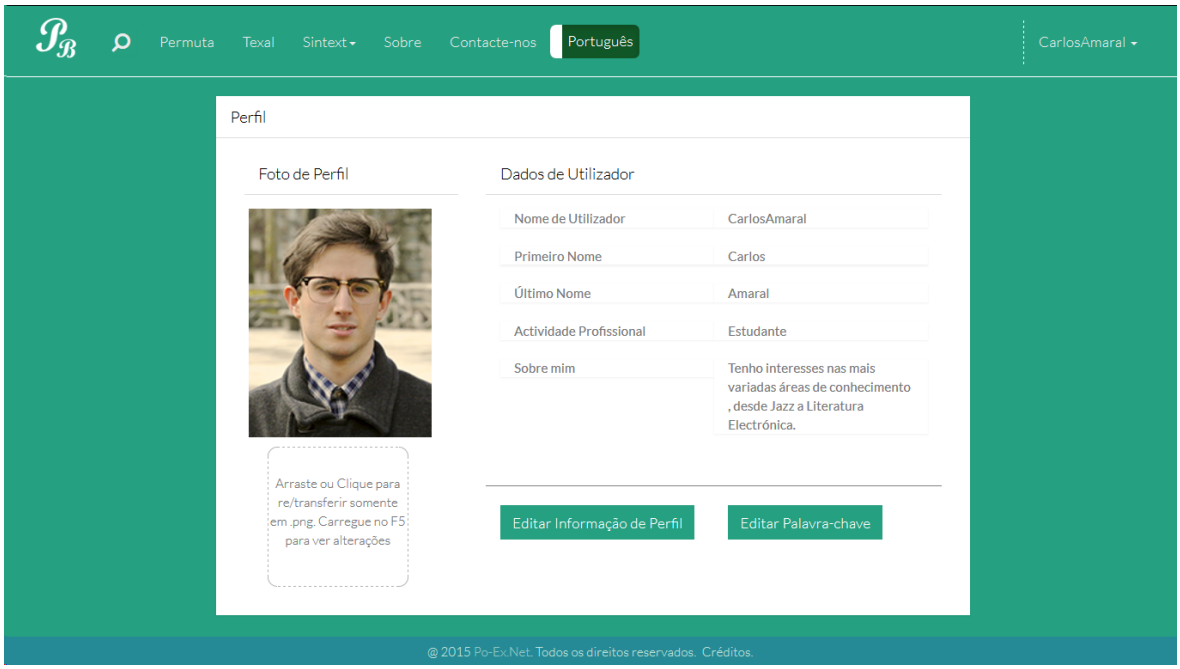


Figura 3.6: Perfil

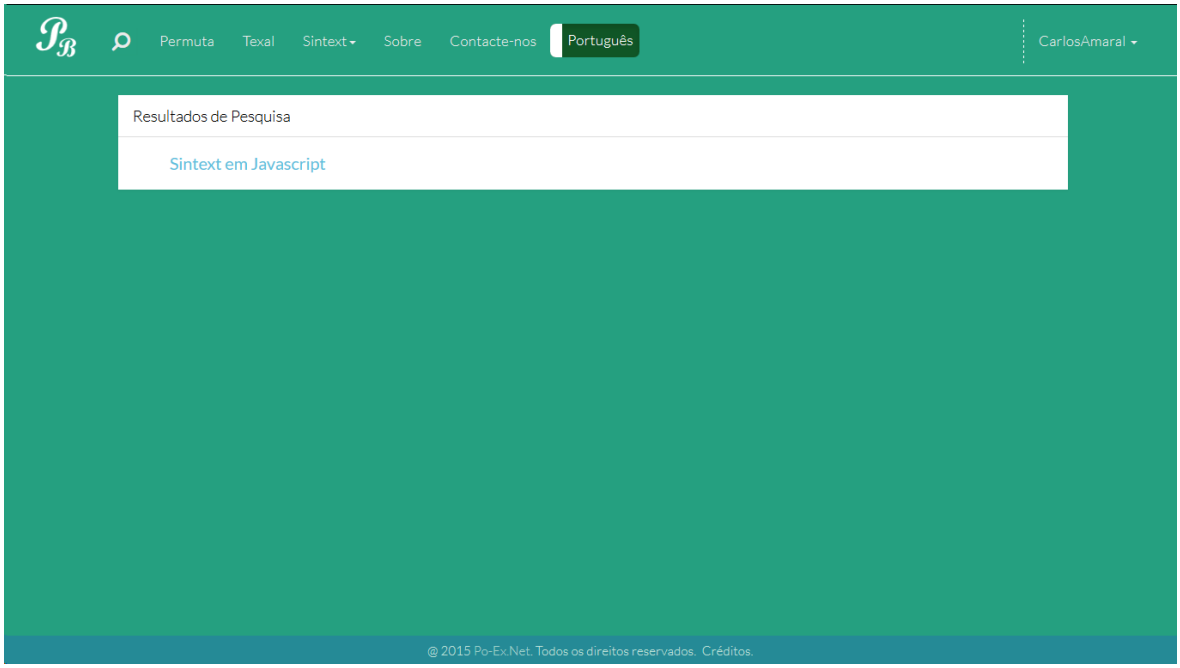


Figura 3.7: Função de Pesquisa

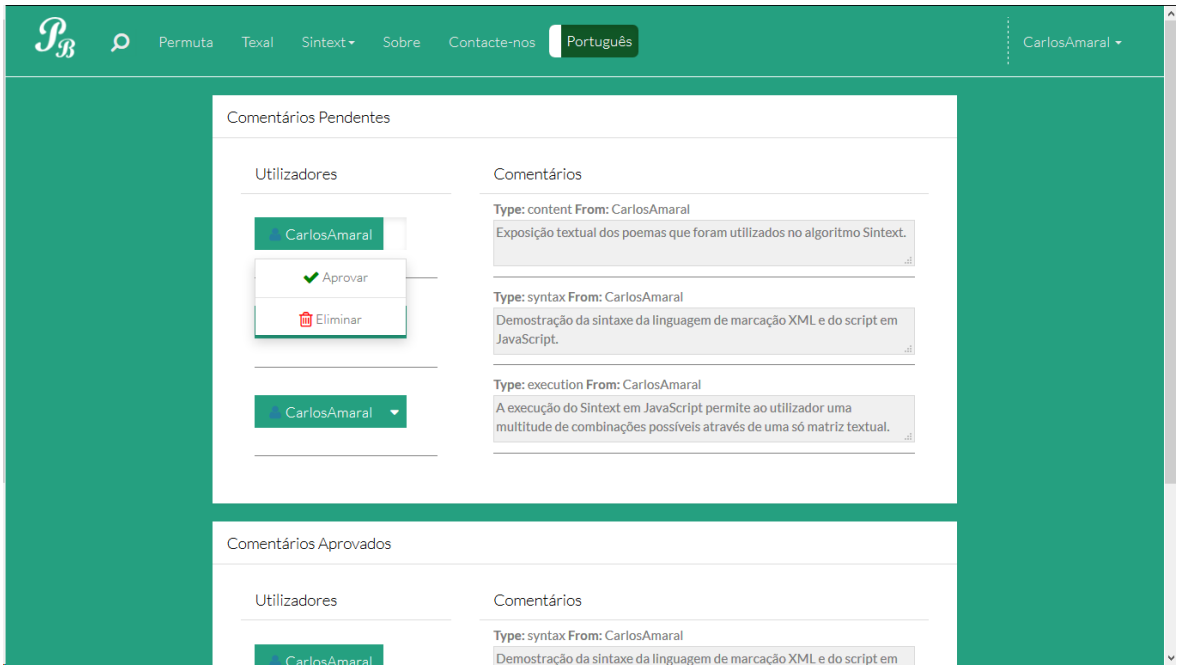


Figura 3.8: Comentários Pendentes

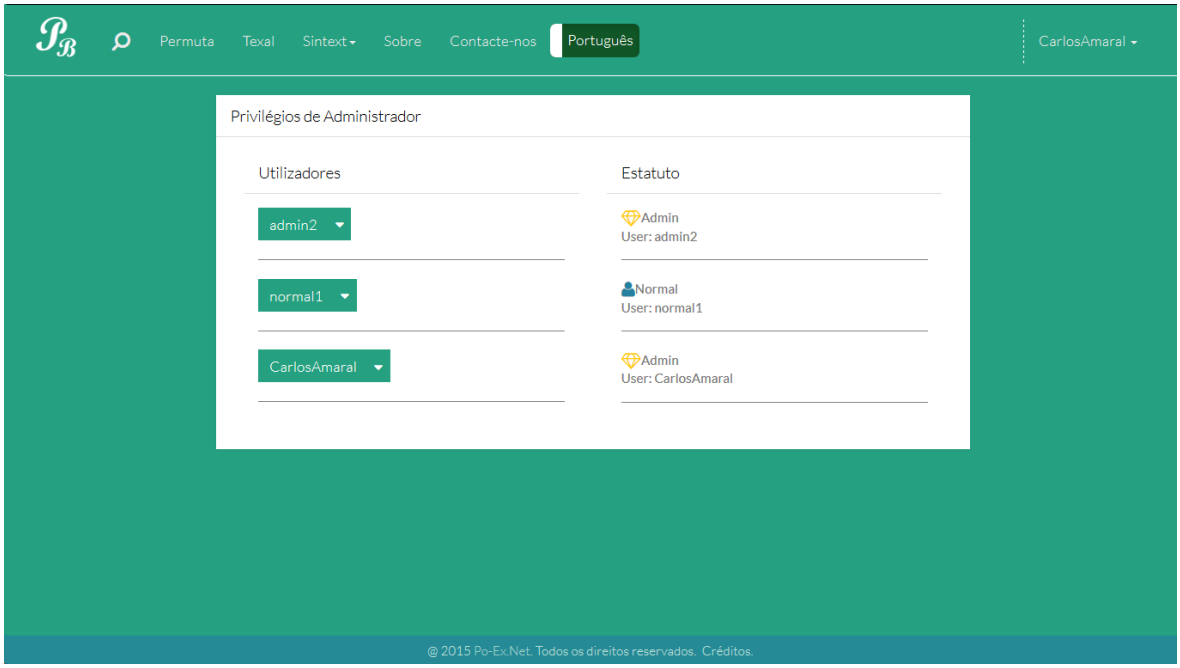


Figura 3.9: Privilégios de Administrador

instruções dadas ao utilizador; 3) mostra a sintaxe por detrás da criação da obra; 4) apresenta uma recensão crítica sobre a obra. No que toca à interactividade, o utilizador Normal pode comentar, sob aprovação de um Administrador, as três últimas secções, e partilhar as páginas no Facebook e Twitter. Por último, o Administrador pode editar os campos de texto dinamicamente, permitindo uma maior facilidade de alteração de conteúdo.

3.6 Conclusões

Neste capítulo, foram demonstrados os requisitos que compreendem o sistema, que serviram de desenho para a arquitectura do sistema, e posterior implementação da Aplicação Web. Não obstante à acentuada curva de aprendizagem da *framework* Laravel e de PHP, a linguagem que lhe está inerente, — serve de especial importância salientar o sucesso da implementação do sistema. A Aplicação Web *Pedro Barbosa: Edição Crítica Digital* serve como modelo para uma Edição Crítica Digital e ficará alojada no Arquivo de Poesia Experimental (PO-EX (dot) Net) em: `http://pedrobarbosa.po-ex.net/`.

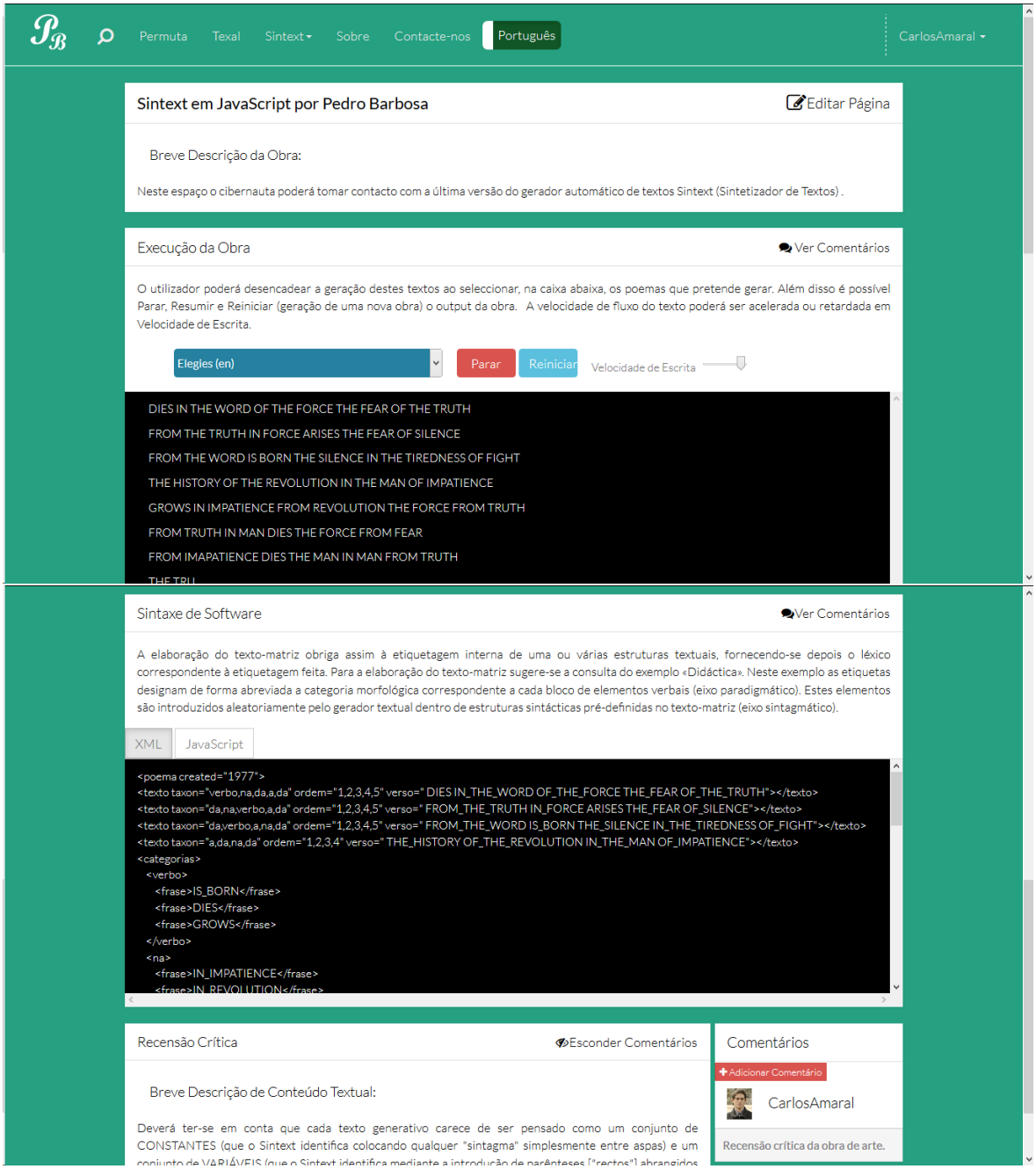


Figura 3.10: Sintext em JavaScript num modelo dividido que aglomera 3 *layers*: 1) execução da obra; 2) código da obra; 3) recensão crítica à obra

Capítulo 4

Conclusão

A literatura electrónica tem vindo gradualmente a marcar as suas pegadas nos estudos dos média, em particular nos média e Humanidades digitais. Este paradigma de apenas duas décadas conta com o apoio da *Electronic Literature Organization* do M.I.T. que irá lançar em breve a terceira colecção de 75 obras literárias (de 600 submissões) nascidas no meio digital. Vimos que, com o advento do computador, foi despoletado um interesse nas Humanidades pelas potencialidades que o computador oferecia. Desde o impacto revolucionário do Padre Roberto Busa, inventor do hipertexto e do primeiro arquivo digital, o laço entre as Humanidades e as Ciências da Computação foi-se solidificando, especialmente após Fortran, a primeira linguagem de programação de alto nível da qual Pedro Barbosa tirou partido, desde então, para o desenvolvimento das suas obras de literatura gerada por computador. Conforme a evolução da Computação e paradoxalmente da evolução dos estudos das Humanidades na Computação, emergiu o paradigma da literatura electrónica: a possibilidade de criação de obras literárias no seio digital que comportam interactividade e multimédia.

Transversalmente à literatura electrónica, uma edição crítica digital de literatura electrónica é uma edição de artefactos digitais, que por isso, levanta outros e novos problemas relacionados com a representatividade destas obras — marcados pelas bases de dados, pela sua contínua transformação, pela variabilidade que lhe está inerente e, consequentemente, afectados pela obsolescência tecnológica que está mais do nunca presente em todas as linguagens e *frameworks* de programação. De certo modo, uma edição crítica digital rejeita os artefactos digitalizados, ou seja, obras que sofreram a conversão do meio analógico para o digital.

No Capítulo *Humanidades na Computação* proporcionou uma visão cronológica da evolução da Computação em paralelo com as Humanidades de forma a compreendermos não só as origens da literatura electrónica e as podermos contextualizar com os artefactos de poesia experimental de Nanni Balestrini, Pedro Barbosa e bpNichol, mas também a relação estabelecida entre ambos os universos e as problemáticas que lhes estão inerentes. A definição de um arquivo digital e de uma edição crítica digital, e de que forma se as obras de literatura electrónica lá são representadas, foram essenciais para a especificação da Aplicação Web *Pedro Barbosa: Edição Crítica Digital*.

Para sustentar esta questão de investigação, o Capítulo *Especificação da Aplicação Web* foram abordados métodos de especificação de requisitos, de desenho de uma arquitectura lógica e posterior uso desses métodos, mais concretamente de linguagens de programação e *software*, para alcançar a solução do problema que fora proposto.

4.1 Satisfação dos Objectivos

Os objectivos do estudo da obra de Pedro Barbosa e a implementação de uma edição crítica digital que suporta as suas obras foram cumpridos. A investigação aqui conduzida permitiu adquirir conhecimento em novas áreas de estudos, enriquecer as que eram familiares, assim como o desenvolvimento de novas práticas e métodos.

4.2 Trabalho Futuro

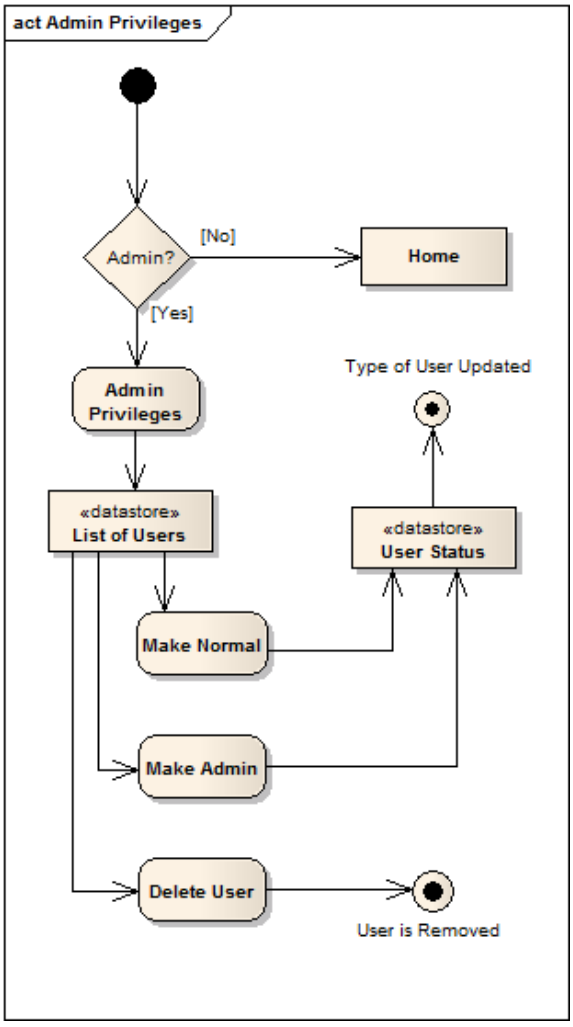
Doravante haverá um esforço na implementação de novas funcionalidades e criação de novo conteúdo que visem, por um lado, melhorar o modelo de edição crítica digital aqui desenvolvido e por outro, enriquecer a nível tecnológico e textual os conteúdos que lá residem. Paralelamente, aplicar o mesmo modelo a autores de obras digitais de literatura electrónica e de outros domínios das Humanidades digitais.

Anexo A

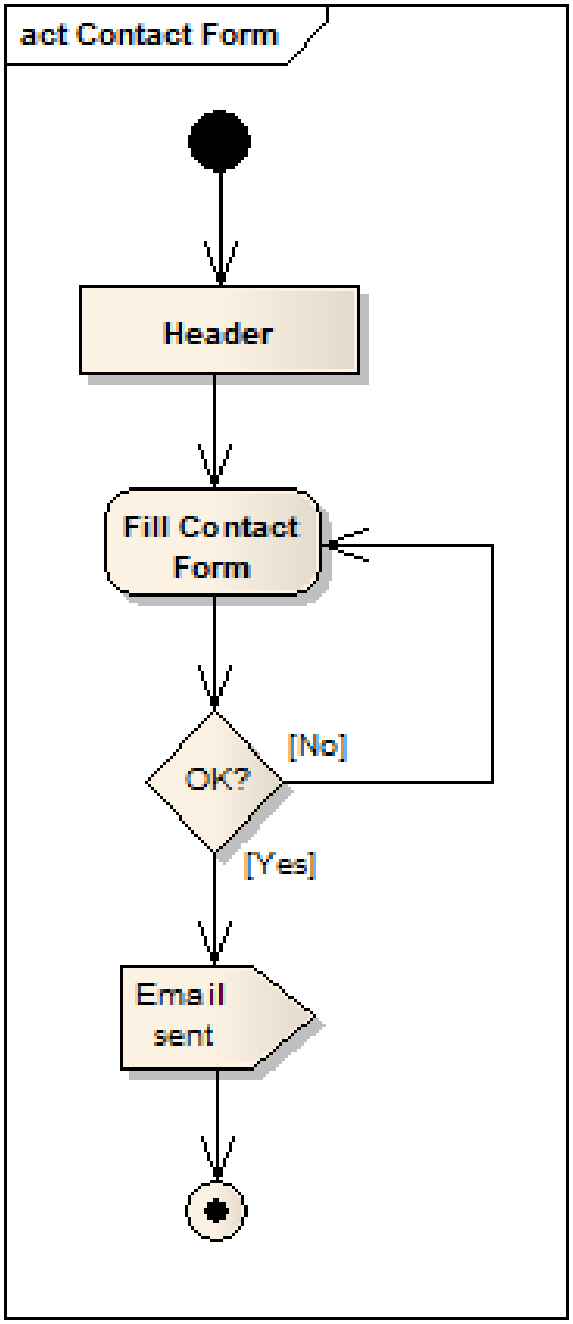
Apêndice

O conteúdo em anexo que se segue contempla diagramas de Narrativas de Utilização que foram substituídos por informação complementar de maior relevância no Capítulo 3.

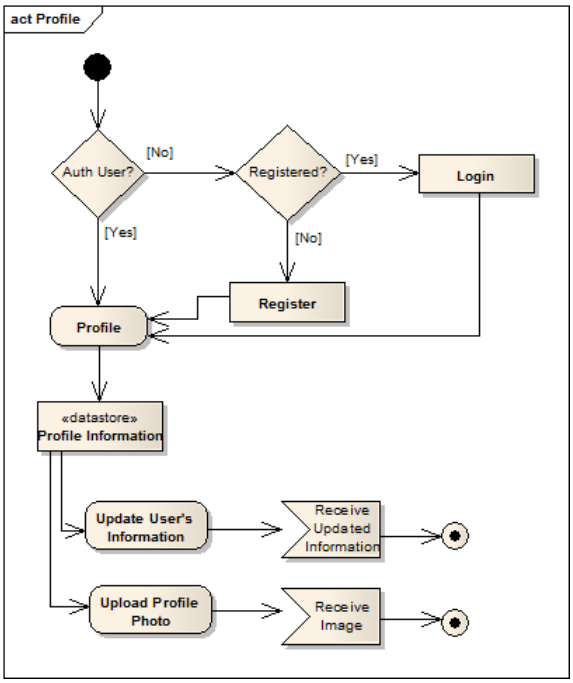
A.1 Narrativas de Utilização



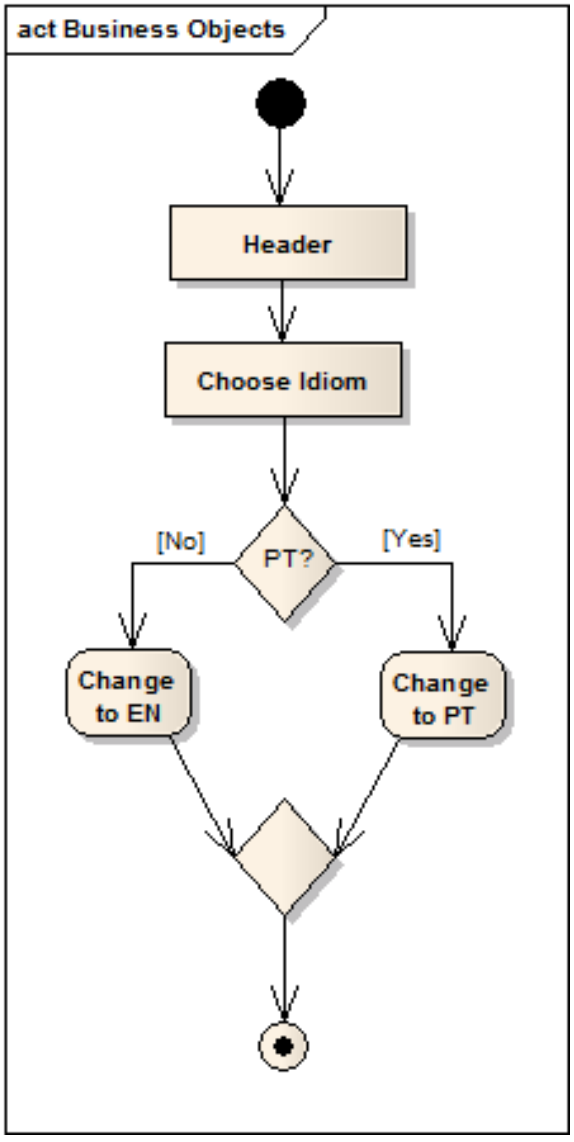
(a) Painel de Administração para alteração de papéis de utilizador



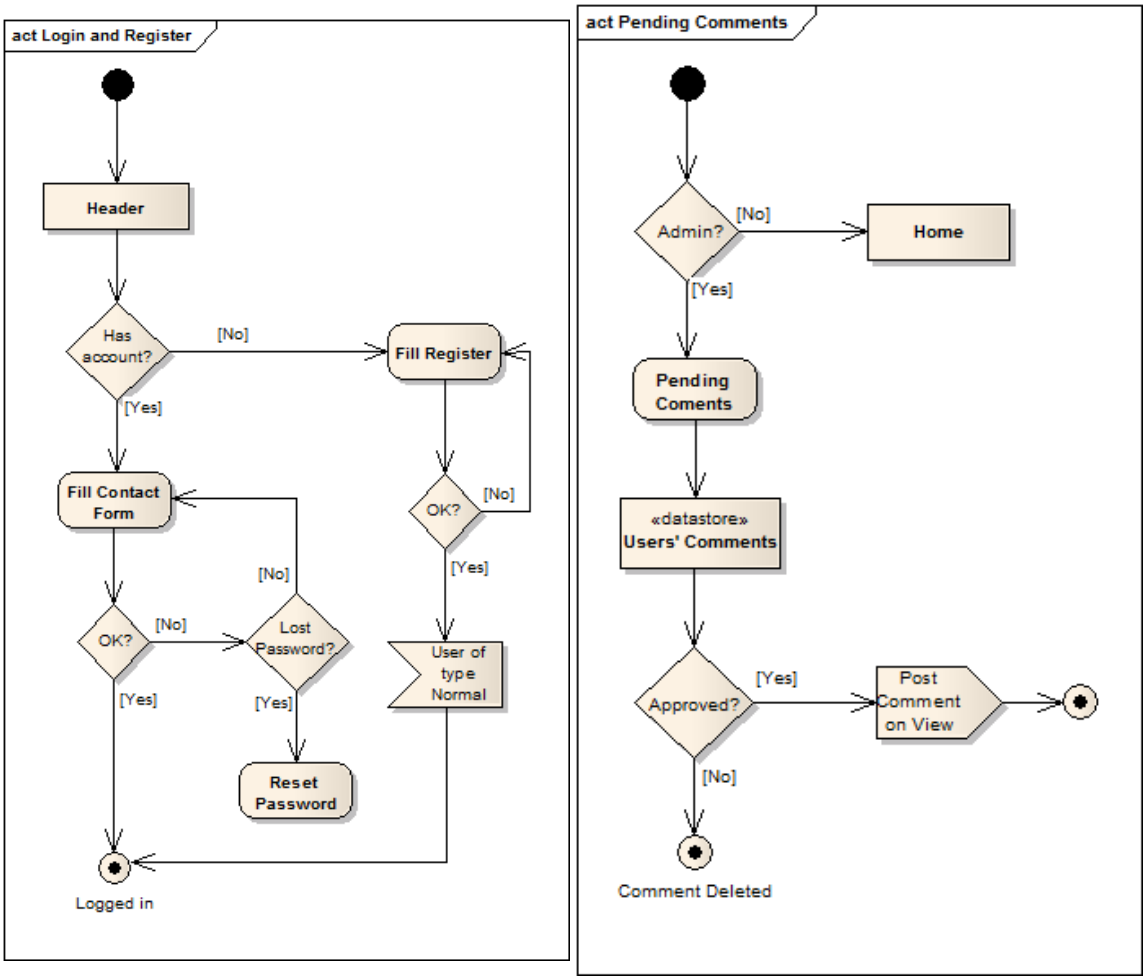
(b) Formulário de Contacto



(a) Perfil de Utilizador



(b) Alteração de Idioma



(a) Autenticação e Registo

(b) Comentários Pendentes

Referências

Abramova, V. and Bernardino, J. (2013). Nosql databases: Mongodb vs cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, C3S2E '13, pages 14–22, New York, NY, USA. ACM.

Ambler, S. (2004). *The object primer : agile modeling-driven development with UML 2.0*. Cambridge University Press, Cambridge, UK New York.

Arden, B., Organick, E., and of Michigan Computing Center, U. (1966). *The Michigan Algorithm Decoder [(the MAD Manual)]*. University of Michigan Computer Center.

Arlow, J. (2005). *UML 2 and the unified process : practical object-oriented analysis and design*. Addison-Wesley, Upper Saddle River, NJ.

Backus, J. W., Beeber, R. J., Best, S., Goldberg, R., Haibt, L. M., Herrick, H. L., Nelson, R. A., Sayre, D., Sheridan, P. B., Stern, H., Ziller, I., Hughes, R. A., and Nutt, R. (1957). The fortran automatic coding system. In *Papers Presented at the February 26-28, 1957, Western Joint Computer Conference: Techniques for Reliability*, IRE-AIEE-ACM '57 (Western), pages 188–198, New York, NY, USA. ACM.

Barbosa, P. (1977). *A Literatura Cibernética 1: autopoemas gerados por computador*. Edições Árvore, Porto.

Barbosa, P. (1980). *A Literatura Cibernética 2: autopoemas gerados por computador*. Edições Árvore, Porto.

Barbosa, P. (1988). *Máquinas Pensantes: aforismos gerados por computador*. Livros Horizonte, Lisboa.

Barbosa, P. (1996a). *A ciberliteratura : criação literaria e computador*. Edições Cosmos, Lisboa.

Barbosa, P. (1996b). *Teoria do Homem Sentado*. Edições Afrontamento, Porto.

Barbosa, P. (1998). Renovação do experimentalismo literário na Literatura Gerada por Computador. *Revista UFP*, 1(2):181–188.

Barbosa, P. (2001). *O Motor Textual*. Edições UFP, Porto.

Barbosa, P. and Cavalheiro, A. (1996). *Teoria do Homem Sentado*. Edições Afrontamento, Porto.

Bayman, P. and Mayer, R. E. (1983). A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements. *Commun. ACM*, 26(9):677–679.

Bean, M. (2015). *Laravel 5 essentials : explore the fundamentals of Laravel, one of the most expressive and robust PHP frameworks available*. Packt Publishing, Birmingham, England.

Benson, Jr., B. W. (1999). Javascript. *SIGPLAN Not.*, 34(4):25–27.

Bergin, T. (1996). *History of programming languages II*. ACM Press Addison-Wesley Pub. Co, New York Reading, Mass.

Berry, D. (2012). *Understanding digital humanities*. Palgrave Macmillan, Houndmills, Basingstoke, Hampshire New York.

Bootz, P. and Funkhouser, C. (2014). *Combinatory and Automatic Text Generation*. The Johns Hopkins Guide to Digital Media. Johns Hopkins University Press.

Brown, J., Trowbridge, T., and Szabo, J. (2009). The poetic metrics of bpnichol. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 933–938.

Brown, J. A. and Trowbridge, T. (2012). Using poetry as a teaching tool for the humanities to learn computer science concepts. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, C3S2E '12, pages 45–52, New York, NY, USA. ACM.

Burdick, A., Drucker, J., Lunenfeld, P., Presner, T., and Schnapp, J. (2012). *Digital humanities*. MIT Press, Cambridge, Mass.

Ceruzzi, P. (2003). *A history of modern computing*. MIT Press, Cambridge, Mass.

Ceruzzi, P. (2012). *Computing a concise history*. MIT Press, Cambridge, Mass.

Christopher, K., Goulish, M., Hixson, L., Jeffery, M., Saner, B., and Walkey, L. (2007). The Goat Island and Performance. http://www.goatislandperformance.org/perf_new.html.

Clark, D. (2006). 88 Constellations for Wittgenstein (to play with the Left Hand). http://www.88constellations.net/.

Clifford, A. (2006). The Sweet Old Etcetera. http://collection.eliterature.org/2/works/clifford_sweet_old_etcetera/sweetweb/sweetoldetc.html.

Clinger, W. (1987). The Scheme Environment. *SIGPLAN Lisp Pointers*, 1(1):27–31.

Clivaz, C. and Hamidovic, D. (2014). *Critical Editionsin the Digital Age*. The Johns Hopkins Guide to Digital Media. Johns Hopkins University Press.

Conner, P. W. (1991). The Beowulf Workstation: One Model of Computer-Assisted Literary Pedagogy. *Literary and Linguistic Computing*, 6(1):50–58.

Corcoran, P. (1974). COCOA: A FORTRAN program for concordance and word-count processing of natural language texts. *Behavior Research Methods & Instrumentation*, 6(6):566–566.

Corporation, D. R. (1975). *All about Remote Computing Services*. A Datapro feature report. The Corporation.

Crane, G. (1996). Building a digital library: The perseus project as a case study in the humanities. In *Proceedings of the First ACM International Conference on Digital Libraries*, DL '96, pages 3–10, New York, NY, USA. ACM.

Crane, G. (2014). Perseus Project. http://www.perseus.tufts.edu/hopper/.

Dahl, O.-J., Myhrhaug, B., and Nygaard, K. (1968). Some features of the simula 67 language. In *Proceedings of the Second Conference on Applications of Simulations*, pages 29–31. Winter Simulation Conference.

Dijkstra, E. W. (1982). *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag New York, Inc., New York, NY, USA.

Ducasse, S., Petton, N., Polito, G., and Cassou, D. (2012). Semantics and Security Issues in JavaScript. *CoRR*, abs/1212.2341.

Duckett, J. (2014). *JavaScript & jQuery : interactive front-end web development*. John Wiley & Sons, Inc, Indianapolis, IN.

Eaves, M., Essick, R., and Viscomi, J. (2015). The William Blake Archive. http://www.blakearchive.org/blake/.

Eden, B. (2005). Associations, Conferences, Discussion Groups/Blogs, Journals, Software/Tools. *Library Technology Reports*, 41(4):13 – 17.

ELO (2015). Electronic Literature Organization. http://eliterature.org.

Ess, C. (2007). *"Revolution? What Revolution?" Successes and Limits of Computing Technologies in Philosophy and Religion*, pages 132–142. Blackwell Publishing Ltd.

Flanders, J. (2005). Detailism, Digital Texts, and the Problem of Pedantry. *TEXT Technology*, pages 41–70.

Fogel, R. (1995). *Time on the cross : the economics of American Negro slavery*. Norton, New York.

Fortran, I. (2015). FORTRAN: The Pioneering Programming Language. http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/fortran/.

Fuller, M. (2008). *Software studies a lexicon*. MIT Press, Cambridge, Mass.

Gasston, P. (2013). *The modern Web multi-device Web development with HTML5, CSS3, and JavaScript*. No Starch Press, San Francisco, CA.

Goldberg, A. and Robson, D. (1983). *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Harris, K. (2014). *Archive*. The Johns Hopkins Guide to Digital Media. Johns Hopkins University Press.

Hayles, K. (2008). *Electronic literature : new horizons for the literary*. University of Notre Dame, Notre Dame, Ind.

Higgins, S. (2011). Digital Curation: The Emergence of a New Discipline. *The International Journal of Digital Curation*, 6(2):78–88.

Hockey, S. (2004). *The History of Humanities Computing*, pages 1–19. Blackwell Publishing Ltd.

Jackson, K. M. (2008). Made to Break: Technology and Obsolescence in America by Giles Slade. *Journal of American Culture*, 31(2):234 – 235.

Jones, S. E. (2006). The William Blake Archive: An Overview. *Literature Compass*, 3(3):409–416.

Joyce, M. (1991). Storyspace as a hypertext system for writers and readers of varying ability. In *Proceedings of the Third Annual ACM Conference on Hypertext*, HYPERTEXT '91, pages 381–387, New York, NY, USA. ACM.

Kurtz, T. E. (1978). BASIC. *SIGPLAN Not.*, 13(8):103–118.

Law, T. M. (2008). Origen's Parallel Bible: Textual Criticism, Apologetics, or Exegesis?. *Journal of Theological Studies*, 59(1):1 – 21.

Lawlor, R. (2015). Delaying obsolescence. *Science and Engineering Ethics*, 21(2):401–427.

Levy, L. S. (1983). A Walk Through AWK. *SIGPLAN Not.*, 18(12):69–85.

Lientz, B. P. (1976). A Comparative Evaluation of Versions of BASIC. *Commun. ACM*, 19(4):175–181.

Liu, Z. H., Hammerschmidt, B., and McMahon, D. (2014). Json data management: Supporting schema-less development in rdbms. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1247–1258, New York, NY, USA. ACM.

Louden, K. (2011). *Programming languages : principles and practice*. Course Technology/Cengage Learning, Boston, MA.

Lutz, T. (2005). Stochastische Texte. *Augenblick 4*.

Maio, A., Bolognesi, G., Dadda, L., and Gregory, T. (2006). *The Work of Roberto Busa SJ: Open Spaces Between Computation and Hermeneutics*. Raccolta delle opere di Roberto Busa. Universidad de Navarra.

Malinconico, S. M. (1984). Planning for Obsolescence. *Library Journal*, 109(3):333.

Manovich, L. (2001). *The language of new media*. MIT Press, Cambridge, Mass.

Marchionini, G. and Crane, G. (1994). Evaluating Hypermedia and Learning: Methods and Results from the Perseus Project. *ACM Trans. Inf. Syst.*, 12(1):5–34.

Masterson, Jr., K. S. (1960). Compilation for Two Computers with NELIAC. *Commun. ACM*, 3(11):607–611.

Mateas, M. and Stern, M. (2005). Façade. http://collection.eliterature.org/2/works/mateas_facade.html/.

McGann, J. (2008). The Rossetti Arquive. http://www.rossettiarchive.org//.

McGann, J. (2010). Electronic Archives and Critical Editing. *Literature Compass*, 7(2):37–42.

Mitchell, J. G., Perlis, A. J., and Van Zoeren, H. R. (1967). Lc2: A language for conversational computing. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, pages 203–214, New York, NY, USA. ACM.

Moghadam, M., Lehman, T., and Zechory, I. (2009). Genius. http://genius.com/.

Morris, J. M. (1995). Experiences with Mosaic for Legacy Projects. *SIGCSE Bull.*, 27(1):292–296.

Morrissey, J. (2007a). The Last Performance. http://thelastperformance.org/title.php.

Morrissey, J. (2007b). The Last Performance [ELO]. http://collection.eliterature.org/2/works/morrissey_lastperformance.html.

Olsen, M. (1989). Textpackv: Text analysis utilities for the personal computer. *Computers and the Humanities*, 23(2):155–160.

Pao, Y. C. (1999). *Engineering analysis : interactive methods and programs with FORTRAN, QuickBASIC, MATLAB, and Mathematica*. CRC Press, Boca Raton, Fla.

Perlis, A., Iturriaga, R., Standish, T., and SCIENCE., C.-M. U. P. P. D. O. C. (1965). *A Preliminary Sketch of Formula ALGOL*. Research paper. Center for the Study of Information Processing, Carnegie Institute of Technology.

Perlis, A. J. (1978). The American Side of the Development of Algol. *SIGPLAN Not.*, 13(8):3–14.

Pressman, J. (2014). *Digital modernism : making it new in new media*. Oxford University Press, New York.

Ramsay, S. (2011). *Reading machines toward an algorithmic criticism*. University of Illinois Press, Urbana.

Raskin, J. (1971). Programming languages for the humanities. *Computers and the Humanities*, 5(3):155–158.

Rauschmayer, A. (2014). *Speaking JavaScript*. O'Reilly Media, Sebastopol, CA.

Rettberg, S. (2014). *Electronic Literature*. The Johns Hopkins Guide to Digital Media. Johns Hopkins University Press.

Sabharwal, A. (2015). *Digital curation in the digital humanities : preserving and promoting archival and special... collections*. Chandos Publishing, S.l.

Sammet, J. (1992). Farewell to Grace Hopper. End of an Era! *Commun. ACM*, 35(4):128–131.

Schwartz, C. (2000). Digital libraries: an overview. *The Journal of Academic Librarianship*, 26(6):85 – 393.

Schwartz, J. I. (1978). The Development of JOVIAL. *SIGPLAN Not.*, 13(8):203–214.

Seiça, l. (2015). Computer Poetry by Silvestre Pestana.

Showstack, R. (2012). Project Gemini online digital archive. *Eos, Transactions American Geophysical Union*, 93(3):31–31.

Simanowski, R. (2010). *Reading moving letters : digital literature in research and teaching : a handbook*. Transcript Verlag Distributed in North America by Transaction Publishers, Bielefeld Piscataway, NJ.

Swierenga, R. P. (1974). Computers and American History: The Impact of the "New" Generation. *The Journal of American History*, 60(4):1045–1070.

Tabbi, J. (2010). Electronic literature as world literature; or, the universality of writing under constraint. *Poetics Today*, 31(1):17 – 50.

Terras, M. (2009). The potential and problems in using high performance computing in the arts and humanities: the researching e-science analysis of census holdings (reach) project. *Digital Humanities Quaterly*, 6(3).

Thomas, W. G. (2004). *Computing and the Historical Imagination*, pages 56–68. Blackwell Publishing Ltd.

Torres, R. (2005). Arquivo Digital da PO.EX. http://po-ex.net/.

Torres, R. (2009). Poemas no meio do caminho. http://collection.eliterature.org/2/works/torres_poemas_caminho.html/.

Wexelblat, R. (1981). *History of programming languages*. Academic Press, New York.

Wirth, N. (1993). Recollections about the development of pascal. In *The Second ACM SIGPLAN Conference on History of Programming Languages*, HOPL-II, pages 333–342, New York, NY, USA. ACM.

Wirth, N. and Weber, H. (1966). EULER: A Generalization of ALGOL and It Formal Definition: Part 1. *Commun. ACM*, 9(1):13–25.

Zielinski, S. (2005). *Variantology : on deep time relations of arts, sciences, and technologies*. W. König, Köln.